

## Load required libraries and datasets

```
In [6]: # Data Wrangling
import numpy as np
import pandas as pd

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Date Functionality
import matplotlib.dates as mdates

# Statistics
from statistics import stdev
from scipy.stats import t

# Remove Warnings
import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: # Loading Data
df1 = pd.read_csv("E:\Deakin\QVI_data.csv")
df1
```

Out[2]:

	LYLTY_CARD_NBR	DATE	STORE_NBR	TXN_ID	PROD_NBR	PROD_NAME
0	1000	2018-10-17		1	1	5 Natural Chip Comnpy SeaSalt175g
1	1002	2018-09-16		1	2	58 Red Rock Deli Chikn&Garlic Aioli 150g
2	1003	2019-03-07		1	3	52 Grain Waves Sour Cream&Chives 210G
3	1003	2019-03-08		1	4	106 Natural ChipCo Hony Soy Chckn175g
4	1004	2018-11-02		1	5	96 WW Original Stacked Chips 160g
...	...	...		...	...	...
264829	2370701	2018-12-08		88	240378	24 Grain Waves Sweet Chilli 210g
264830	2370751	2018-10-01		88	240394	60 Kettle Tortilla ChpsFeta&Garlic 150g
264831	2370961	2018-10-24		88	240480	70 Tyrrells Crisps Lightly Salted 165g
264832	2370961	2018-10-27		88	240481	65 Old El Paso Salsa Dip Chnky Tom Ht300g
264833	2373711	2018-12-14		88	241815	16 Smiths Crinkle Chips Salt & Vinegar 330g

264834 rows × 12 columns



## To Check Missing Data

In [3]: `df1.isnull().sum()`

```
Out[3]: LYLTY_CARD_NBR      0
        DATE            0
        STORE_NBR       0
        TXN_ID          0
        PROD_NBR        0
        PROD_NAME       0
        PROD_QTY         0
        TOT_SALES        0
        PACK_SIZE        0
        BRAND           0
        LIFESTAGE        0
        PREMIUM_CUSTOMER 0
        dtype: int64
```

## Select control stores

The client has selected store numbers 77, 86 and 88 as trial stores and want control stores to be established stores that are operational for the entire observation period. We would want to match trial stores to control stores that are similar to the trial store prior to the trial period of Feb 2019 in terms of :

- Monthly overall sales revenue
- Monthly number of customers
- Monthly number of transactions per customer

```
In [4]: ##### Create a month ID in format yyyyymm
df1['YEARMONTH'] = [ ''.join(x.split('-')[0:2]) for x in df1.DATE]
df1['YEARMONTH'] = pd.to_numeric(df1['YEARMONTH'])
df1['YEARMONTH']
```

```
Out[4]: 0      201810
1      201809
2      201903
3      201903
4      201811
...
264829 201812
264830 201810
264831 201810
264832 201810
264833 201812
Name: YEARMONTH, Length: 264834, dtype: int64
```

## For each store and month calculate

- Total sales.
- Number of customers.
- Transactions per customer.
- Chips per customer.
- Average price per unit.

In [5]: # Total Sales for each store and month.

```
Total_Sales= df1.groupby(['STORE_NBR', 'YEARMONTH']).TOT_SALES.sum()
Total_Sales
```

Out[5]: STORE\_NBR YEARMONTH

1	201807	206.9
	201808	176.1
	201809	278.8
	201810	188.1
	201811	192.6
		...
272	201902	395.5
	201903	442.3
	201904	445.1
	201905	314.6
	201906	312.1

Name: TOT\_SALES, Length: 3169, dtype: float64

In [6]: # Number of customers for each store and month.

```
no_Customers = df1.groupby(['STORE_NBR', 'YEARMONTH']).LYLTY_CARD_NBR.nunique()
no_Customers
```

Out[6]: STORE\_NBR YEARMONTH

1	201807	49
	201808	42
	201809	59
	201810	44
	201811	46
		..
272	201902	45
	201903	50
	201904	54
	201905	34
	201906	34

Name: LYLTY\_CARD\_NBR, Length: 3169, dtype: int64

In [7]: # Transactions per customer for each store and month.

```
trans_per_customer = df1.groupby(['STORE_NBR', 'YEARMONTH']).TXN_ID.nunique()
trans_per_customer
```

Out[7]: STORE\_NBR YEARMONTH

1	201807	1.061224
	201808	1.023810
	201809	1.050847
	201810	1.022727
	201811	1.021739
		...
272	201902	1.066667
	201903	1.060000
	201904	1.018519
	201905	1.176471
	201906	1.088235

Length: 3169, dtype: float64

In [8]: # Chips per customer for each store and month.

```
chips_per_customer = df1.groupby(['STORE_NBR', 'YEARMONTH']).PROD_QTY.sum()/
chips_per_customer
```

Out[8]: STORE\_NBR YEARMONTH

	STORE_NBR	YEARMONTH	
1		201807	1.192308
		201808	1.255814
		201809	1.209677
		201810	1.288889
		201811	1.212766
			...
272		201902	1.895833
		201903	1.905660
		201904	1.909091
		201905	1.775000
		201906	1.891892

Length: 3169, dtype: float64

In [9]: # Average price per unit for each store and month.

```
average_price = df1.groupby(['STORE_NBR', 'YEARMONTH']).TOT_SALES.sum()/
df1.average_price
```

Out[9]: STORE\_NBR YEARMONTH

	STORE_NBR	YEARMONTH	
1		201807	3.337097
		201808	3.261111
		201809	3.717333
		201810	3.243103
		201811	3.378947
			...
272		201902	4.346154
		201903	4.379208
		201904	4.239048
		201905	4.430986
		201906	4.458571

Length: 3169, dtype: float64

In [10]: # Concatenate into a new dataframe 'measureOverTime'

```
df2 = [Total_Sales, no_Customers, trans_per_customer, chips_per_customer, ave
measureOverTime = pd.concat(df2, join = 'outer', axis = 1)
measureOverTime
```

Out[10]:

		TOT_SALES	LYLTY_CARD_NBR	0	1
	STORE_NBR	YEARMONTH			
1	201807	206.9	49	1.061224	1.192308
	201808	176.1	42	1.023810	1.255814
	201809	278.8	59	1.050847	1.209677
	201810	188.1	44	1.022727	1.288889
	201811	192.6	46	1.021739	1.212766
	...	...	...	...	...
272	201902	395.5	45	1.066667	1.895833
	201903	442.3	50	1.060000	1.905660
	201904	445.1	54	1.018519	1.909091
	201905	314.6	34	1.176471	1.775000
	201906	312.1	34	1.088235	1.891892

3169 rows × 5 columns

In [11]: # Rename the columns

```
measureOverTime.rename(columns = {'TOT_SALES': 'Total Sales', 'LYLTY_CARD_NBR': 'Loyalty Card NBR'}  
measureOverTime.head(15)
```

Out[11]:

		Total Sales	no_Customers	trans_per_customer	chips_per_custo
	STORE_NBR	YEARMONTH			
1	201807	206.9	49	1.061224	1.192
	201808	176.1	42	1.023810	1.251
	201809	278.8	59	1.050847	1.209
	201810	188.1	44	1.022727	1.288
	201811	192.6	46	1.021739	1.212
	201812	189.6	42	1.119048	1.212
	201901	154.8	35	1.028571	1.166
	201902	225.4	52	1.057692	1.187
	201903	192.9	45	1.088889	1.183
	201904	192.9	42	1.023810	1.321
	201905	221.4	46	1.108696	1.294
	201906	174.1	42	1.000000	1.267
2	201807	150.8	39	1.051282	1.121
	201808	193.8	39	1.102564	1.279
	201809	154.4	36	1.027778	1.108



```
In [12]: measureOverTime.reset_index(inplace = True)  
measureOverTime
```

Out[12]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3169 rows × 7 columns

In [13]: *#### Filter to the pre-trial period and stores with full observation periods*

```
df3 = pd.pivot_table(df1, index = 'STORE_NBR', columns = 'YEARMONTH', values
```

```
df3
```

Out[13]: YEARMONTH 201807 201808 201809 201810 201811 201812 201901 201902

STORE_NBR		201807	201808	201809	201810	201811	201812	201901	201902
1	52.0	43.0	62.0	45.0	47.0	47.0	36.0	55.0	
2	41.0	43.0	37.0	43.0	40.0	38.0	45.0	32.0	
3	138.0	134.0	119.0	119.0	118.0	129.0	121.0	139.0	
4	160.0	151.0	138.0	155.0	139.0	133.0	168.0	102.0	
5	120.0	112.0	125.0	107.0	111.0	125.0	118.0	106.0	
...	...	...	...	...	...	...	...	...	...
268	52.0	54.0	34.0	48.0	51.0	43.0	38.0	37.0	
269	139.0	132.0	124.0	148.0	136.0	133.0	144.0	133.0	
270	139.0	154.0	126.0	119.0	133.0	149.0	155.0	125.0	
271	129.0	101.0	114.0	114.0	122.0	117.0	120.0	102.0	
272	52.0	48.0	36.0	51.0	45.0	47.0	50.0	48.0	

272 rows × 12 columns



In [14]: # Store numbers that do not have full observation periods

```
null_stores = df3[df3.isnull().any(axis=1)].index.tolist()
null_stores
```

Out[14]: [11, 31, 44, 76, 85, 92, 117, 193, 206, 211, 218, 252]

In [15]: # Filter out the null stores

```
measureOverTime = measureOverTime[~measureOverTime['STORE_NBR'].isin(null_sto
measureOverTime
```

Out[15]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3120 rows × 7 columns

In [16]:

```
# Filter to pre-trial period before 201902
preTrialMeasures = measureOverTime[measureOverTime['YEARMONTH'] < 201902]
preTrialMeasures
```

Out[16]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3159	272	201809	304.7	32	1.125000	
3160	272	201810	430.6	44	1.136364	
3161	272	201811	376.2	41	1.097561	
3162	272	201812	403.9	47	1.000000	
3163	272	201901	423.0	46	1.086957	

1820 rows × 7 columns

Now we need to work out a way of ranking how similar each potential control store is to the trial store. We can calculate how correlated the performance of each store is to the trial store.

```
In [17]: #Let's write a function for this so that we don't have to calculate this for
##### Create a function to calculate correlation for a measure

def calculateCorrelation(inputTable, metricCol, storeComparison):
    calcCorrTable = pd.DataFrame({'Store1': [], 'Store2': [], 'corr_measure': []})
    x = inputTable.loc[inputTable['STORE_NBR'] == trial_store, metricCol]
    x.reset_index(drop = True, inplace = True)
    storeNumbers = inputTable['STORE_NBR'].unique()
    for i in storeNumbers:
        y = inputTable.loc[inputTable['STORE_NBR'] == i, metricCol]
        y.reset_index(drop = True, inplace = True)
        calcCorrTable = calcCorrTable.append({'Store1': trial_store, 'Store2': i, 'corr_measure': np.corrcoef(x, y)[0][1]})
    return calcCorrTable
```

Apart from correlation, we can also calculate a standardised metric based on the absolute difference between the trial store's performance and each control store's performance.

```
In [18]: def calculateMagnitudeDistance(inputTable, metricCol, storeComparison):
    calcDistTable = pd.DataFrame({'Store1': [], 'Store2': [], 'Magnitude' : []})
    x = inputTable.loc[inputTable['STORE_NBR'] == trial_store, metricCol]
    x.reset_index(drop = True, inplace = True)
    storeNumbers = inputTable['STORE_NBR'].unique()
    for i in storeNumbers:
        y = inputTable.loc[inputTable['STORE_NBR'] == i, metricCol]
        y.reset_index(drop = True, inplace = True)
        z = abs(x-y)
        A = np.mean(1-(z-min(z))/(max(z)-min(z)))
        calcDistTable = calcDistTable.append({'Store1': trial_store, 'Store2': i, 'Magnitude': A})
    return calcDistTable
```

We'll select control stores based on how similar monthly total sales in dollar amounts and monthly number of customers are to the trial stores.

## Selecting control store for trial store 77

```
In [19]: trial_store = 77

corr_Sales = calculateCorrelation(preTrialMeasures, 'Total Sales', trial_store)
corr_Customers = calculateCorrelation(preTrialMeasures, 'no_Customers', trial_store)

# Compute magnitude with trial store 77
magnitude_Sales = calculateMagnitudeDistance(preTrialMeasures, 'Total Sales', trial_store)
magnitude_Customers = calculateMagnitudeDistance(preTrialMeasures, 'no_Customers', trial_store)
```

```
In [20]: score_Sales = pd.concat([corr_Sales, magnitude_Sales['Magnitude']], axis = 1)
score_Sales
```

Out[20]:

	<b>Store1</b>	<b>Store2</b>	<b>corr_measure</b>	<b>Magnitude</b>
<b>0</b>	77.0	1.0	0.075218	0.408163
<b>1</b>	77.0	2.0	-0.263079	0.590119
<b>2</b>	77.0	3.0	0.806644	0.522914
<b>3</b>	77.0	4.0	-0.263300	0.644934
<b>4</b>	77.0	5.0	-0.110652	0.516320
...	...	...	...	...
<b>255</b>	77.0	268.0	0.344757	0.429787
<b>256</b>	77.0	269.0	-0.315730	0.559099
<b>257</b>	77.0	270.0	0.315430	0.591547
<b>258</b>	77.0	271.0	0.355487	0.341091
<b>259</b>	77.0	272.0	0.117622	0.523631

260 rows × 4 columns

```
In [21]: score_Customers = pd.concat([corr_Customers, magnitude_Customers['Magnitude']]
```

```
In [22]: score_Customers
```

Out[22]:

	<b>Store1</b>	<b>Store2</b>	<b>corr_measure</b>	<b>Magnitude</b>
<b>0</b>	77.0	1.0	0.322168	0.663866
<b>1</b>	77.0	2.0	-0.572051	0.471429
<b>2</b>	77.0	3.0	0.834207	0.489796
<b>3</b>	77.0	4.0	-0.295639	0.498258
<b>4</b>	77.0	5.0	0.370659	0.512605
...	...	...	...	...
<b>255</b>	77.0	268.0	0.369517	0.571429
<b>256</b>	77.0	269.0	-0.474293	0.484472
<b>257</b>	77.0	270.0	-0.131259	0.536680
<b>258</b>	77.0	271.0	0.019629	0.328571
<b>259</b>	77.0	272.0	0.223217	0.545455

260 rows × 4 columns

```
In [23]: # Add an additional column which calculates the weighted average

corr_weight = 0.5
score_Sales['scoreSales'] = corr_weight * score_Sales['corr_measure'] + (1 - score_Customers['scoreCustomers']) = corr_weight * score_Customers['corr_measu
```

```
In [24]: score_Sales.set_index(['Store1', 'Store2'], inplace = True)
score_Customers.set_index(['Store1', 'Store2'], inplace = True)

score_Control = pd.concat([score_Sales['scoreSales'], score_Customers['scoreC
score_Control
```

Out[24]:

		scoreSales	scoreCustomers
	Store1	Store2	
77.0	1.0	0.241691	0.493017
	2.0	0.163520	-0.050311
	3.0	0.664779	0.662002
	4.0	0.190817	0.101310
	5.0	0.202834	0.441632
	...	...	...
268.0	0.387272	0.470473	
269.0	0.121684	0.005090	
270.0	0.453489	0.202710	
271.0	0.348289	0.174100	
272.0	0.320626	0.384336	

260 rows × 2 columns

```
In [25]: # Add a new column 'finalControlScore' to 'score_Control'.

score_Control['finalControlScore'] = 0.5 * (score_Control['scoreSales'] + sco
score_Control
```

Out[25]:

		scoreSales	scoreCustomers	finalControlScore
Store1	Store2			
77.0	1.0	0.241691	0.493017	0.367354
	2.0	0.163520	-0.050311	0.056604
	3.0	0.664779	0.662002	0.663390
	4.0	0.190817	0.101310	0.146064
	5.0	0.202834	0.441632	0.322233
	...	...	...	...
268.0		0.387272	0.470473	0.428872
269.0		0.121684	0.005090	0.063387
270.0		0.453489	0.202710	0.328100
271.0		0.348289	0.174100	0.261195
272.0		0.320626	0.384336	0.352481

260 rows × 3 columns

In [26]: `score_Control.sort_values(by = 'finalControlScore', ascending = False).head()`

Out[26]:

		scoreSales	scoreCustomers	finalControlScore
Store1	Store2			
77.0	233.0	0.697290	0.816607	0.756949
	71.0	0.789497	0.663123	0.726310
	84.0	0.656972	0.715000	0.685986
	119.0	0.636046	0.729729	0.682887
	115.0	0.708347	0.645155	0.676751

- For Trial Store 77, control store is 233.

**Now that we have found a control store, let's check visually if the drivers are indeed similar in the period before the trial.**

**We'll look at total sales first.**

In [27]: `control_store = 233``pastSales = preTrialMeasures`

```

store_type = []

for i in pastSales['STORE_NBR']:
    if i == trial_store:
        store_type.append('Trial Store')
    elif i == control_store:
        store_type.append('Control Store')
    else:
        store_type.append('Other Stores')

pastSales['store_type'] = store_type
pastSales.head()

```

Out[27]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_per_cu
0	1	201807	206.9	49	1.061224	1
1	1	201808	176.1	42	1.023810	1
2	1	201809	278.8	59	1.050847	1
3	1	201810	188.1	44	1.022727	1
4	1	201811	192.6	46	1.021739	1

In [28]: `pastSales['TransactionMonth'] = pd.to_datetime(pastSales['YEARMONTH'].astype(str).str[:6])`

Out[28]:

```

0      2018-07-01
1      2018-08-01
2      2018-09-01
3      2018-10-01
4      2018-11-01
       ...
3159   2018-09-01
3160   2018-10-01
3161   2018-11-01
3162   2018-12-01
3163   2019-01-01
Name: TransactionMonth, Length: 1820, dtype: datetime64[ns]

```

In [29]: `pastSales['Month'] = pd.DatetimeIndex(pastSales['TransactionMonth']).month_name`

Out[29]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3159	272	201809	304.7	32	1.125000	
3160	272	201810	430.6	44	1.136364	
3161	272	201811	376.2	41	1.097561	
3162	272	201812	403.9	47	1.000000	
3163	272	201901	423.0	46	1.086957	

1820 rows × 10 columns

In [30]:

```
controlPlot = pastSales.loc[pastSales['store_type'] == 'Control Store', ['Month']]
controlPlot.set_index('Month', inplace = True)

trialPlot = pastSales.loc[pastSales['store_type'] == 'Trial Store', ['Month']]
trialPlot.set_index('Month', inplace = True)

otherPlot = pastSales.loc[pastSales['store_type'] == 'Other Stores', ['Month']]
otherPlot = pd.DataFrame(otherPlot.groupby('Month')['Total Sales'].mean())

# Renaming Column Names
controlPlot.rename(columns = {'Total Sales': 'Control Store Sales'}, inplace = True)
trialPlot.rename(columns = {'Total Sales': 'Trial Store Sales'}, inplace = True)
otherPlot.rename(columns = {'Total Sales': 'Other Stores Sales'}, inplace = True)

# Concatenate
combinePlot = pd.concat([controlPlot, trialPlot, otherPlot], axis = 1)

combinePlot
```

Out[30]:

	Control Store Sales	Trial Store Sales	Other Stores Sales
<b>Month</b>			
<b>Jul</b>	290.7	296.8	638.004651
<b>Aug</b>	285.9	255.5	610.223450
<b>Sep</b>	228.6	225.2	620.198450
<b>Oct</b>	185.7	204.5	635.314729
<b>Nov</b>	211.6	245.3	618.864341
<b>Dec</b>	279.8	267.3	648.453876
<b>Jan</b>	177.5	204.4	628.684496

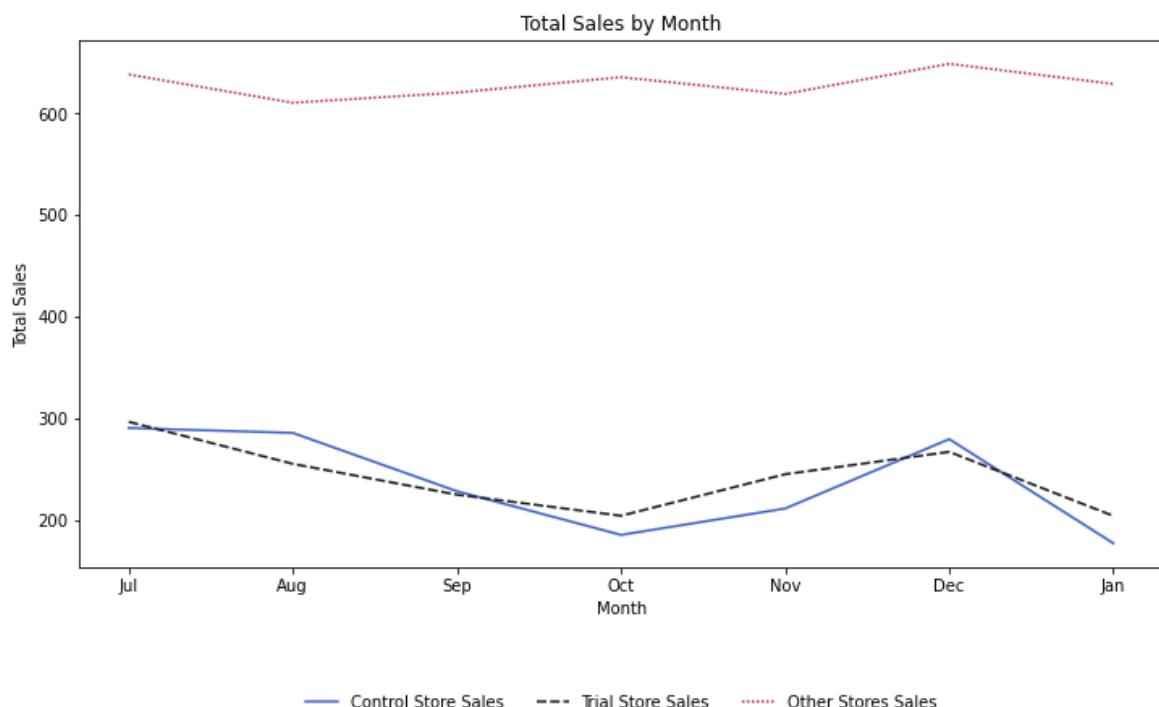
In [31]: # Plot total Sales for all 3 types of stores

```

plt.figure(figsize = (12, 6))
plt.title('Total Sales by Month')
ax = sns.lineplot(data=combinePlot, palette="icefire")

plt.ylabel("Total Sales")
sns.move_legend(
    ax, "lower center",
    bbox_to_anchor=(0.5, -.3),
    ncol=3,
    frameon=False,
)

```



**Visual checks on customer count trends by comparing the trial store to the control store and other stores.**

```
In [32]: control_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Control Store']
control_Customer_Plot.set_index('Month', inplace = True)

trial_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Trial Store',
trial_Customer_Plot.set_index('Month', inplace = True)

other_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Other Stores']
other_Customer_Plot = pd.DataFrame(other_Customer_Plot.groupby('Month')['no_Customers'].sum())

# Renaming Column Names
control_Customer_Plot.rename(columns = {'no_Customers': 'Control Store Customers'}, inplace = True)
trial_Customer_Plot.rename(columns = {'no_Customers': 'Trial Store Customers'}, inplace = True)
other_Customer_Plot.rename(columns = {'no_Customers': 'Other Stores Customers'}, inplace = True)

# Concatenate
combine_Customer_Plot = pd.concat([control_Customer_Plot, trial_Customer_Plot,
                                     other_Customer_Plot])

combine_Customer_Plot
```

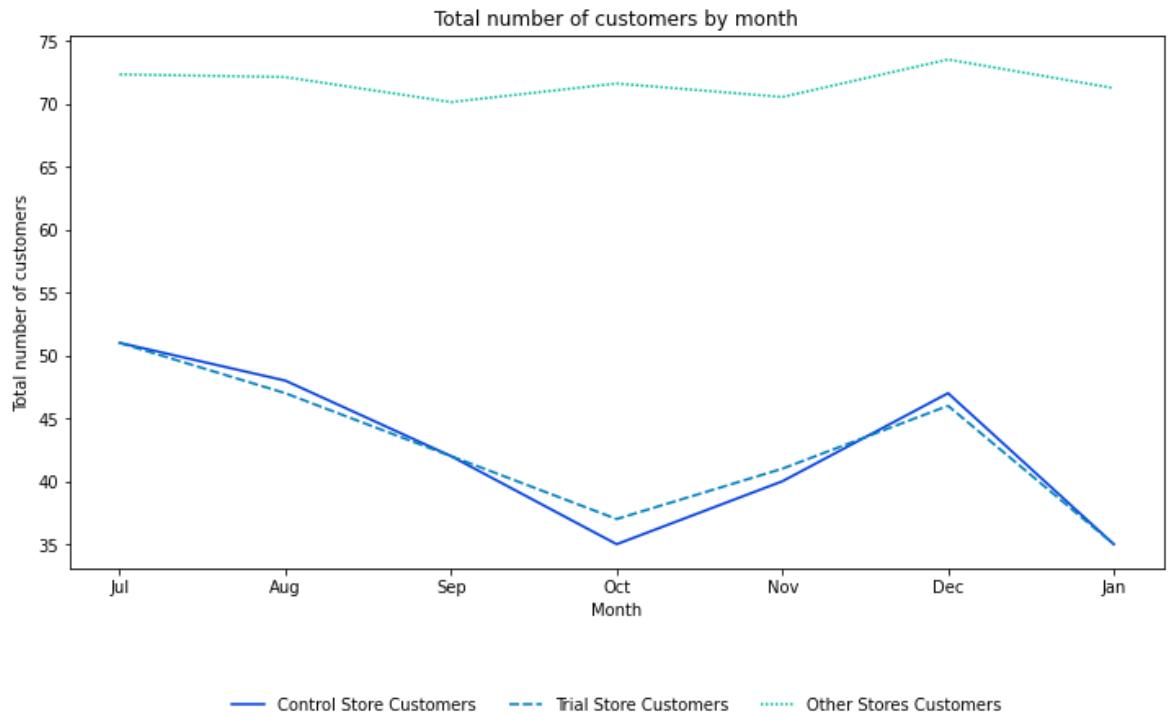
Out[32]:

	Control Store Customers	Trial Store Customers	Other Stores Customers
<b>Month</b>			
<b>Jul</b>	51	51	72.333333
<b>Aug</b>	48	47	72.120155
<b>Sep</b>	42	42	70.131783
<b>Oct</b>	35	37	71.608527
<b>Nov</b>	40	41	70.534884
<b>Dec</b>	47	46	73.515504
<b>Jan</b>	35	35	71.240310

In [33]: # Plot total number of customers for all 3 types of stores

```
plt.figure(figsize = (12, 6))
plt.title('Total number of customers by month')
ax1 = sns.lineplot(data=combine_Customer_Plot, palette="winter")

plt.ylabel("Total number of customers")
sns.move_legend(
    ax1, "lower center",
    bbox_to_anchor=(0.5, - .3),
    ncol=3,
    frameon=False,
)
```



## Assessment of trial

The trial period goes from the start of February 2019 to April 2019. We now want to see if there has been an uplift in overall chip sales.

In [34]: preTrialMeasures

Out[34]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3159	272	201809	304.7	32	1.125000	
3160	272	201810	430.6	44	1.136364	
3161	272	201811	376.2	41	1.097561	
3162	272	201812	403.9	47	1.000000	
3163	272	201901	423.0	46	1.086957	

1820 rows × 10 columns

In [35]: 

```
# Scale pre-trial control sales to match pre-trial trial store sales
```

```
trial_sales_sum = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Tri'
control_sales_sum = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'C'
scalingFactorForControlSales = trial_sales_sum / control_sales_sum
scalingFactorForControlSales
```

Out[35]: 1.023617303289553

In [36]: 

```
# Apply the scaling factor
```

```
measureOverTimeSales = measureOverTime
scaledControlSales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'
scaledControlSales['controlSales'] = scaledControlSales['Total Sales'] * scal
scaledControlSales
```

Out[36]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
2699	233	201807	290.7	51	1.058824	
2700	233	201808	285.9	48	1.041667	
2701	233	201809	228.6	42	1.071429	
2702	233	201810	185.7	35	1.028571	
2703	233	201811	211.6	40	1.025000	
2704	233	201812	279.8	47	1.063830	
2705	233	201901	177.5	35	1.000000	
2706	233	201902	244.0	45	1.044444	
2707	233	201903	199.1	40	1.025000	
2708	233	201904	158.6	30	1.066667	
2709	233	201905	344.4	57	1.087719	
2710	233	201906	221.0	41	1.000000	



In [37]: # Add a new column 'TransactionMonth' to 'scaledControlSales'

```
scaledControlSales['TransactionMonth'] = pd.to_datetime(scaledControlSales['Y']  
scaledControlSales
```

Out[37]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
2699	233	201807	290.7	51	1.058824	
2700	233	201808	285.9	48	1.041667	
2701	233	201809	228.6	42	1.071429	
2702	233	201810	185.7	35	1.028571	
2703	233	201811	211.6	40	1.025000	
2704	233	201812	279.8	47	1.063830	
2705	233	201901	177.5	35	1.000000	
2706	233	201902	244.0	45	1.044444	
2707	233	201903	199.1	40	1.025000	
2708	233	201904	158.6	30	1.066667	
2709	233	201905	344.4	57	1.087719	
2710	233	201906	221.0	41	1.000000	



Now that we have comparable sales figures for the control store, we can calculate the percentage difference between the scaled control sales and the trial store's sales during the trial period.

In [38]: `measureOverTime`

Out[38]:

	<b>STORE_NBR</b>	<b>YEARMONTH</b>	<b>Total Sales</b>	<b>no_Customers</b>	<b>trans_per_customer</b>	<b>chips_pe</b>
<b>0</b>	1	201807	206.9	49	1.061224	
<b>1</b>	1	201808	176.1	42	1.023810	
<b>2</b>	1	201809	278.8	59	1.050847	
<b>3</b>	1	201810	188.1	44	1.022727	
<b>4</b>	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
<b>3164</b>	272	201902	395.5	45	1.066667	
<b>3165</b>	272	201903	442.3	50	1.060000	
<b>3166</b>	272	201904	445.1	54	1.018519	
<b>3167</b>	272	201905	314.6	34	1.176471	
<b>3168</b>	272	201906	312.1	34	1.088235	

3120 rows × 7 columns

In [39]: `percentageDiff = scaledControlSales[['YEARMONTH', 'controlSales']]`  
`percentageDiff.reset_index(drop = True, inplace = True)`

In [40]: `trial_sales = measureOverTime.loc[measureOverTime['STORE_NBR'] == trial_store]`  
`trial_sales.reset_index(drop = True, inplace = True)`

`percentageDiff = pd.concat([percentageDiff, trial_sales], axis = 1)`  
`percentageDiff.rename(columns = {'Total Sales': 'Trial_Sales'}, inplace = True)`  
`percentageDiff`

Out[40]:

	YEARMONTH	controlSales	Trial_Sales
0	201807	297.565550	296.8
1	201808	292.652187	255.5
2	201809	233.998916	225.2
3	201810	190.085733	204.5
4	201811	216.597421	245.3
5	201812	286.408121	267.3
6	201901	181.692071	204.4
7	201902	249.762622	235.0
8	201903	203.802205	278.5
9	201904	162.345704	263.5
10	201905	352.533799	299.3
11	201906	226.219424	264.7

In [41]:

```
percentageDiff['percentageDiff'] = abs(percentageDiff.controlSales - percentageDiff.Trial_Sales) / percentageDiff.controlSales
```

Out[41]:

	YEARMONTH	controlSales	Trial_Sales	percentageDiff
0	201807	297.565550	296.8	0.002573
1	201808	292.652187	255.5	0.126950
2	201809	233.998916	225.2	0.037602
3	201810	190.085733	204.5	0.075830
4	201811	216.597421	245.3	0.132516
5	201812	286.408121	267.3	0.066716
6	201901	181.692071	204.4	0.124980
7	201902	249.762622	235.0	0.059107
8	201903	203.802205	278.5	0.366521
9	201904	162.345704	263.5	0.623080
10	201905	352.533799	299.3	0.151003
11	201906	226.219424	264.7	0.170103

As our null hypothesis is that the trial period is the same as the pre-trial period, let's take the standard deviation based on the scaled percentage difference in the pre-trial period

```
In [42]: stdDev = stdev(percentageDiff.loc[percentageDiff['YEARMONTH'] < 201902, 'perc
stdDev
```

Out[42]: 0.049940762641425544

```
In [43]: # Note that there are 8 months in the pre-trial period , Hence 8 - 1 = 7 degr
degreesOfFreedom = 7
```

```
In [44]: ##### We will test with a null hypothesis of there being 0 difference between
percentageDiff['tValue'] = (percentageDiff['percentageDiff'] - 0) / stdDev
percentageDiff.loc[(percentageDiff['YEARMONTH'] < 201905 ) & (percentageDiff[
```

```
Out[44]: 7    1.183534
8    7.339116
9    12.476373
Name: tValue, dtype: float64
```

```
In [45]: ##### Find the 95th percentile of the t distribution with the degreesOfFreedom
```

```
t95 = t.isf(0.05, degreesOfFreedom)
print('95th percentile of the t distribution with the degrees Of Freedom 7
```

95th percentile of the t distribution with the degrees Of Freedom 7 - 1.8945  
786050613054

- We can observe that the t-value is much larger than the 95th percentile value of the t-distribution for March and April - i.e. the increase in sales in the trial store in March and April is statistically greater than in the control store.

```
In [46]: measureOverTimeSales = measureOverTime
measureOverTimeSales
```

Out[46]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3120 rows × 7 columns



In [47]:

```
measureOverTimeSales['TransactionMonth'] = pd.to_datetime(measureOverTimeSales)
measureOverTimeSales
```

Out[47]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3120 rows × 8 columns



In [48]:

```
#### Trial and control store total sales
```

```
pastSales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] == tri  
pastSales.set_index('TransactionMonth', inplace = True)  
pastSales.rename(columns = {'Total Sales': 'Trial Sales'}, inplace = True)  
pastSales
```

Out[48]:

**Trial Sales**

TransactionMonth	Trial Sales
2018-07-01	296.8
2018-08-01	255.5
2018-09-01	225.2
2018-10-01	204.5
2018-11-01	245.3
2018-12-01	267.3
2019-01-01	204.4
2019-02-01	235.0
2019-03-01	278.5
2019-04-01	263.5
2019-05-01	299.3
2019-06-01	264.7

In [49]:

```
#### Control store 95th percentile  
  
controlSales = scaledControlSales.loc[:, ['TransactionMonth', 'controlSales']]  
controlSales.set_index('TransactionMonth', inplace = True)  
controlSales.rename(columns = {'controlSales': 'Control Sales'}, inplace = True)  
controlSales
```

Out[49]:

**Control Sales**

TransactionMonth	Control Sales
2018-07-01	297.565550
2018-08-01	292.652187
2018-09-01	233.998916
2018-10-01	190.085733
2018-11-01	216.597421
2018-12-01	286.408121
2019-01-01	181.692071
2019-02-01	249.762622
2019-03-01	203.802205
2019-04-01	162.345704
2019-05-01	352.533799
2019-06-01	226.219424

```
In [50]: ##### Control store 95th percentile
controlSales['Control store 95th percentile'] = controlSales['Control Sales']

##### Control store 5th percentile

controlSales['Control store 5th percentile'] = controlSales['Control Sales']

controlSales
```

Out[50]:

TransactionMonth	Control Sales	Control store 95th percentile	Control store 5th percentile
2018-07-01	297.565550	327.286851	267.844249
2018-08-01	292.652187	321.882734	263.421640
2018-09-01	233.998916	257.371084	210.626747
2018-10-01	190.085733	209.071786	171.099680
2018-11-01	216.597421	238.231502	194.963341
2018-12-01	286.408121	315.015001	257.801241
2019-01-01	181.692071	199.839753	163.544390
2019-02-01	249.762622	274.709294	224.815950
2019-03-01	203.802205	224.158280	183.446130
2019-04-01	162.345704	178.561041	146.130368
2019-05-01	352.533799	387.745413	317.322186
2019-06-01	226.219424	248.814565	203.624283

In [51]: trialAssessment = pd.merge(controlSales, pastSales, left\_index = True, right\_index = True)

Out[51]:

TransactionMonth	Control Sales	Control store 95th percentile	Control store 5th percentile	Trial Sales
2018-07-01	297.565550	327.286851	267.844249	296.8
2018-08-01	292.652187	321.882734	263.421640	255.5
2018-09-01	233.998916	257.371084	210.626747	225.2
2018-10-01	190.085733	209.071786	171.099680	204.5
2018-11-01	216.597421	238.231502	194.963341	245.3
2018-12-01	286.408121	315.015001	257.801241	267.3
2019-01-01	181.692071	199.839753	163.544390	204.4
2019-02-01	249.762622	274.709294	224.815950	235.0
2019-03-01	203.802205	224.158280	183.446130	278.5
2019-04-01	162.345704	178.561041	146.130368	263.5
2019-05-01	352.533799	387.745413	317.322186	299.3
2019-06-01	226.219424	248.814565	203.624283	264.7

- Let's create a more visual version of this by plotting the sales of the control store, the sales of the trial stores and the 95th percentile value of sales of the control store.

```
In [52]: plt.figure(figsize = (14,8))
ax2 = sns.lineplot(data=trialAssessment, palette = 'magma')

sns.move_legend(
    ax2, "lower center",
    bbox_to_anchor=(0.5, -.2),
    ncol=3,
    frameon=False,
)

plt.title('Total Sales by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
plt.ylim((0, 450))

plt.axvspan(*mdates.datestr2num(['2019-02-01', '2019-04-01']), color = 'orange')
plt.grid()
```



The trial period is highlighted by the orange strip.

The results show that the trial in store 77 is significantly different to its control store in the trial period as the trial store performance lies outside the 5% to 95% confidence interval of the control store in two of the three trial months.

Let's have a look at assessing this for number of customers as well.

```
In [53]: trial_customers = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Trial']
control_customers = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Control']
scalingFactorForControlCust = trial_customers / control_customers
scalingFactorForControlCust
```

Out[53]: 1.0033557046979866

```
In [54]: measureOverTimeCusts = measureOverTime
measureOverTimeCusts
```

Out[54]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_per_trans
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3120 rows × 8 columns



```
In [55]: scaledControlCustomers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR'] == 272]
scaledControlCustomers
```

Out[55]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
2699	233	201807	290.7	51	1.058824	
2700	233	201808	285.9	48	1.041667	
2701	233	201809	228.6	42	1.071429	
2702	233	201810	185.7	35	1.028571	
2703	233	201811	211.6	40	1.025000	
2704	233	201812	279.8	47	1.063830	
2705	233	201901	177.5	35	1.000000	
2706	233	201902	244.0	45	1.044444	
2707	233	201903	199.1	40	1.025000	
2708	233	201904	158.6	30	1.066667	
2709	233	201905	344.4	57	1.087719	
2710	233	201906	221.0	41	1.000000	

In [56]: `scaledControlCustomers['Control_Customers'] = scaledControlCustomers['no_Customers'] / scaledControlCustomers['Total Sales'] * 100`

Out[56]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
2699	233	201807	290.7	51	1.058824	
2700	233	201808	285.9	48	1.041667	
2701	233	201809	228.6	42	1.071429	
2702	233	201810	185.7	35	1.028571	
2703	233	201811	211.6	40	1.025000	
2704	233	201812	279.8	47	1.063830	
2705	233	201901	177.5	35	1.000000	
2706	233	201902	244.0	45	1.044444	
2707	233	201903	199.1	40	1.025000	
2708	233	201904	158.6	30	1.066667	
2709	233	201905	344.4	57	1.087719	
2710	233	201906	221.0	41	1.000000	

In [57]: `# Calculate the percentage difference between scaled control customers. and total sales`

```
percentageDiff = scaledControlCustomers[['YEARMONTH', 'Control_Customers']]
percentageDiff.reset_index(drop = True, inplace = True)
percentageDiff
```

Out[57]:

	YEARMONTH	Control_Customers
0	201807	51.171141
1	201808	48.161074
2	201809	42.140940
3	201810	35.117450
4	201811	40.134228
5	201812	47.157718
6	201901	35.117450
7	201902	45.151007
8	201903	40.134228
9	201904	30.100671
10	201905	57.191275
11	201906	41.137584

In [58]:

```
trial_Customers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR']]
trial_Customers.reset_index(drop = True, inplace = True)
trial_Customers
```

Out[58]:

```
0    51
1    47
2    42
3    37
4    41
5    46
6    35
7    45
8    50
9    47
10   55
11   41
Name: no_Customers, dtype: int64
```

In [59]:

```
percentageDiff = pd.concat([percentageDiff, trial_Customers], axis = 1)
percentageDiff.rename(columns = {'no_Customers': 'Trial_Customers'}, inplace = True)
percentageDiff
```

Out[59]:

	YEARMONTH	Control_Customers	Trial_Customers
0	201807	51.171141	51
1	201808	48.161074	47
2	201809	42.140940	42
3	201810	35.117450	37
4	201811	40.134228	41
5	201812	47.157718	46
6	201901	35.117450	35
7	201902	45.151007	45
8	201903	40.134228	50
9	201904	30.100671	47
10	201905	57.191275	55
11	201906	41.137584	41

In [60]: percentageDiff['percentageDiff'] = abs(percentageDiff.Control\_Customers - per  
percentageDiff

Out[60]:

	YEARMONTH	Control_Customers	Trial_Customers	percentageDiff
0	201807	51.171141	51	0.003344
1	201808	48.161074	47	0.024108
2	201809	42.140940	42	0.003344
3	201810	35.117450	37	0.053607
4	201811	40.134228	41	0.021572
5	201812	47.157718	46	0.024550
6	201901	35.117450	35	0.003344
7	201902	45.151007	45	0.003344
8	201903	40.134228	50	0.245819
9	201904	30.100671	47	0.561427
10	201905	57.191275	55	0.038315
11	201906	41.137584	41	0.003344

**Let's again see if the difference is significant visually!**In [61]: stdDev = stdev(percentageDiff.loc[percentageDiff['YEARMONTH'] < 201902, 'perc  
degreesOfFreedom = 7

As our null hypothesis is that the trial period is the same as the pre-trial period, let's take the standard deviation based on the scaledpercentage difference in the pre-trial period

```
In [62]: percentageDiff['T_value'] = (percentageDiff['percentageDiff'] - 0) / stdDev
percentageDiff.loc[(percentageDiff['YEARMONTH'] > 201901) & (percentageDiff['
```

```
Out[62]: 7    0.183352
          8    13.476388
          9    30.778725
Name: T_value, dtype: float64
```

```
In [63]: ##### 95th percentile of the t distribution with degreesOfFreedom = 7

t95 = t.isf(0.05, degreesOfFreedom)
print('95th percentile of the t distribution with the degrees Of Freedom 7')
```

```
95th percentile of the t distribution with the degrees Of Freedom 7      - 1.8945
786050613054
```

We Can see that the t-value is much greater than the 95th percentile of the t distribution

```
In [64]: controlCustomers = scaledControlCustomers.loc[:, ['TransactionMonth', 'Control']]
controlCustomers.set_index('TransactionMonth', inplace = True)
controlCustomers
```

```
Out[64]:
```

### Control\_Customers

TransactionMonth	
2018-07-01	51.171141
2018-08-01	48.161074
2018-09-01	42.140940
2018-10-01	35.117450
2018-11-01	40.134228
2018-12-01	47.157718
2019-01-01	35.117450
2019-02-01	45.151007
2019-03-01	40.134228
2019-04-01	30.100671
2019-05-01	57.191275
2019-06-01	41.137584

```
In [65]: trialCustomers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR'] =
trialCustomers.set_index('TransactionMonth', inplace = True)
```

```
trialCustomers.rename(columns = {'no_Customers': 'Trial Customers'}, inplace=True)
```

Out[65]:

**Trial Customers**

TransactionMonth	
2018-07-01	51
2018-08-01	47
2018-09-01	42
2018-10-01	37
2018-11-01	41
2018-12-01	46
2019-01-01	35
2019-02-01	45
2019-03-01	50
2019-04-01	47
2019-05-01	55
2019-06-01	41

In [66]:

```
#### Control store 95th percentile
controlCustomers['Control 95th % confidence Interval'] = controlCustomers['Control 95th % confidence Interval'].round(0)
controlCustomers['Control 5th % confidence Interval'] = controlCustomers['Control 5th % confidence Interval'].round(0)

# Merge trial store as well to the control store dataframe.
trial_77_Customer_Assessment = pd.merge(controlCustomers, trialCustomers, left_index=True, right_index=True)
```

In [67]:

trial\_77\_Customer\_Assessment

Out[67]:

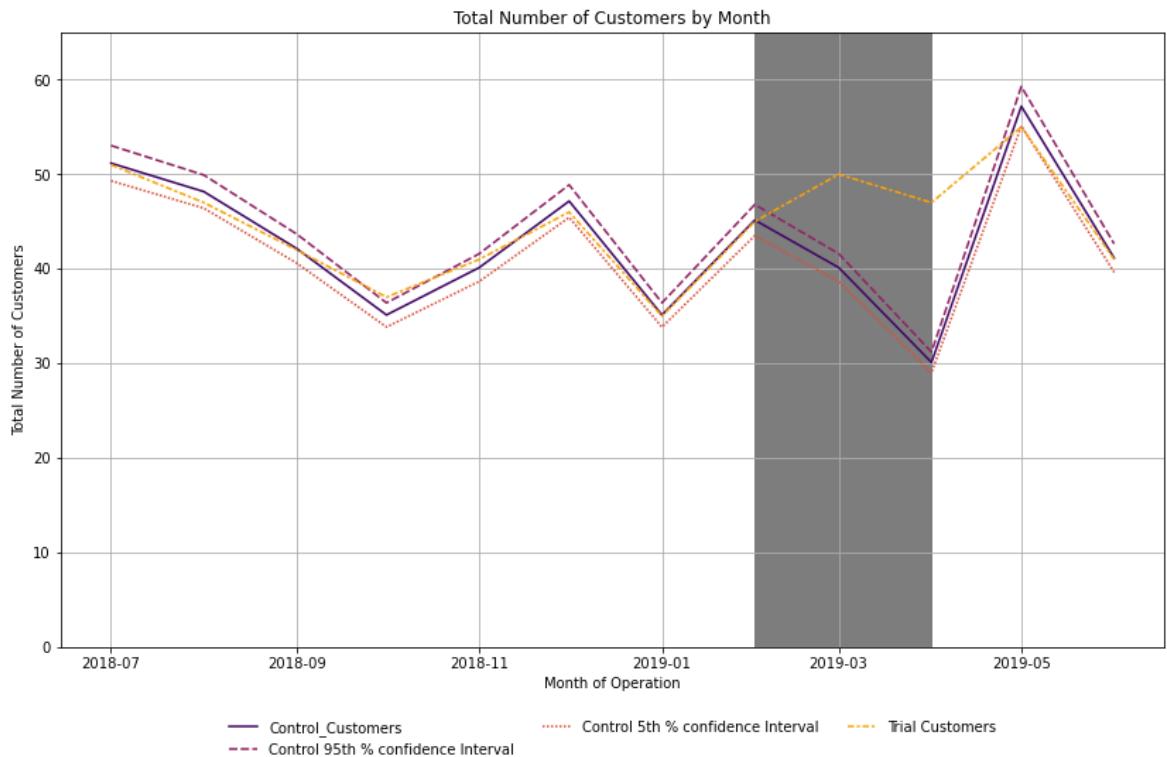
TransactionMonth	Control_Customers	Control 95th % confidence Interval	Control 5th % confidence Interval	Trial Customers
2018-07-01	51.171141	53.037941	49.304341	51
2018-08-01	48.161074	49.918062	46.404086	47
2018-09-01	42.140940	43.678304	40.603575	42
2018-10-01	35.117450	36.398587	33.836313	37
2018-11-01	40.134228	41.598385	38.670071	41
2018-12-01	47.157718	48.878102	45.437334	46
2019-01-01	35.117450	36.398587	33.836313	35
2019-02-01	45.151007	46.798183	43.503830	45
2019-03-01	40.134228	41.598385	38.670071	50
2019-04-01	30.100671	31.198789	29.002554	47
2019-05-01	57.191275	59.277699	55.104852	55
2019-06-01	41.137584	42.638345	39.636823	41

```
In [68]: plt.figure(figsize = (14,8))
ax3 = sns.lineplot(data=trial_77_Customer_Assessment, palette = 'inferno')

sns.move_legend(
    ax3, "lower center",
    bbox_to_anchor=(0.5, -.2),
    ncol=3,
    frameon=False,
)

plt.title('Total Number of Customers by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Number of Customers')
plt.ylim((0, 65))

plt.axvspan(*mdates.datestr2num(['2019-02-01', '2019-04-01']), color = 'grey')
plt.grid()
```



Let's repeat finding the control store and assessing the impact of the trial for each of the other two trial stores.

## Trial store 86

```
In [69]: trial_store = 86

corr_Sales = calculateCorrelation(preTrialMeasures, 'Total Sales', trial_stor
corr_Customers = calculateCorrelation(preTrialMeasures, 'no_Customers', trial

# Compute magnitude with trial store 86
magnitude_Sales = calculateMagnitudeDistance(preTrialMeasures, 'Total Sales',
magnitude_Customers = calculateMagnitudeDistance(preTrialMeasures, 'no_Customers', trial_store)

In [70]: score_Sales = pd.concat([corr_Sales, magnitude_Sales['Magnitude']], axis = 1)
score_Sales
```

Out[70]:

	<b>Store1</b>	<b>Store2</b>	<b>corr_measure</b>	<b>Magnitude</b>
<b>0</b>	86.0	1.0	0.445632	0.488334
<b>1</b>	86.0	2.0	-0.403835	0.321131
<b>2</b>	86.0	3.0	-0.261284	0.507515
<b>3</b>	86.0	4.0	-0.039035	0.635654
<b>4</b>	86.0	5.0	0.235159	0.579835
...	...	...	...	...
<b>255</b>	86.0	268.0	-0.452182	0.291930
<b>256</b>	86.0	269.0	0.697055	0.480267
<b>257</b>	86.0	270.0	-0.730679	0.517014
<b>258</b>	86.0	271.0	0.527637	0.565665
<b>259</b>	86.0	272.0	0.004926	0.583840

260 rows × 4 columns

In [71]: `score_Customers = pd.concat([corr_Customers, magnitude_Customers['Magnitude']])`In [72]: `# Add an additional column which calculates the weighted average`

```

corr_weight = 0.5
score_Sales['scoreSales'] = corr_weight * score_Sales['corr_measure'] + (1 - 
score_Customers['scoreCustomers']) = corr_weight * score_Customers['corr_measu

```

In [73]: `score_Sales.set_index(['Store1', 'Store2'], inplace = True)
score_Customers.set_index(['Store1', 'Store2'], inplace = True)`

```

score_Control = pd.concat([score_Sales['scoreSales'], score_Customers['scoreC
score_Control

```

Out[73]:

**scoreSales scoreCustomers**

Store1	Store2	scoreSales	scoreCustomers
<b>86.0</b>	<b>1.0</b>	0.466983	0.498018
	<b>2.0</b>	-0.041352	0.171205
	<b>3.0</b>	0.123116	0.104620
	<b>4.0</b>	0.298309	0.184103
	<b>5.0</b>	0.407497	0.230528
	...	...	...
<b>268.0</b>	-0.080126	0.266027	
<b>269.0</b>	0.588661	0.241523	
<b>270.0</b>	-0.106832	-0.058237	
<b>271.0</b>	0.546651	0.432804	
<b>272.0</b>	0.294383	0.139863	

260 rows × 2 columns

In [74]: *# Add a new column 'finalControlScore' to 'score\_Control'.*

```
score_Control['finalControlScore'] = 0.5 * (score_Control['scoreSales'] + sco  
score_Control
```

Out[74]:

		scoreSales	scoreCustomers	finalControlScore
Store1	Store2			
86.0	1.0	0.466983	0.498018	0.482500
	2.0	-0.041352	0.171205	0.064927
	3.0	0.123116	0.104620	0.113868
	4.0	0.298309	0.184103	0.241206
	5.0	0.407497	0.230528	0.319013
	...	...	...	...
	268.0	-0.080126	0.266027	0.092950
	269.0	0.588661	0.241523	0.415092
	270.0	-0.106832	-0.058237	-0.082535
	271.0	0.546651	0.432804	0.489727
	272.0	0.294383	0.139863	0.217123

260 rows × 3 columns

In [75]: `score_Control.sort_values(by = 'finalControlScore', ascending = False).head()`

Out[75]:

		scoreSales	scoreCustomers	finalControlScore
Store1	Store2			
86.0	155.0	0.808106	0.733343	0.770724
	109.0	0.697120	0.742532	0.719826
	114.0	0.631393	0.663384	0.647389
	225.0	0.601841	0.684356	0.643099
	138.0	0.593296	0.660565	0.626930

- For Trial Store 86, control store is 155.

```
In [76]: control_store = 155
pastSales = preTrialMeasures

store_type = []

for i in pastSales['STORE_NBR']:
    if i == trial_store:
        store_type.append('Trial Store')
    elif i == control_store:
        store_type.append('Control Store')
    else:
        store_type.append('Other Stores')
```

```
pastSales['store_type'] = store_type
pastSales.head()
```

Out[76]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_per_cu
0	1	201807	206.9	49	1.061224	1
1	1	201808	176.1	42	1.023810	1
2	1	201809	278.8	59	1.050847	1
3	1	201810	188.1	44	1.022727	1
4	1	201811	192.6	46	1.021739	1

In [77]:

```
controlPlot = pastSales.loc[pastSales['store_type'] == 'Control Store', ['Month']]
controlPlot.set_index('Month', inplace = True)

trialPlot = pastSales.loc[pastSales['store_type'] == 'Trial Store', ['Month']]
trialPlot.set_index('Month', inplace = True)

otherPlot = pastSales.loc[pastSales['store_type'] == 'Other Stores', ['Month']]
otherPlot = pd.DataFrame(otherPlot.groupby('Month')['Total Sales'].mean())

# Renaming Column Names
controlPlot.rename(columns = {'Total Sales': 'Control Store Sales'}, inplace = True)
trialPlot.rename(columns = {'Total Sales': 'Trial Store Sales'}, inplace = True)
otherPlot.rename(columns = {'Total Sales': 'Other Stores Sales'}, inplace = True)

# Concatenate
combinePlot = pd.concat([controlPlot, trialPlot, otherPlot], axis = 1)

combinePlot
```

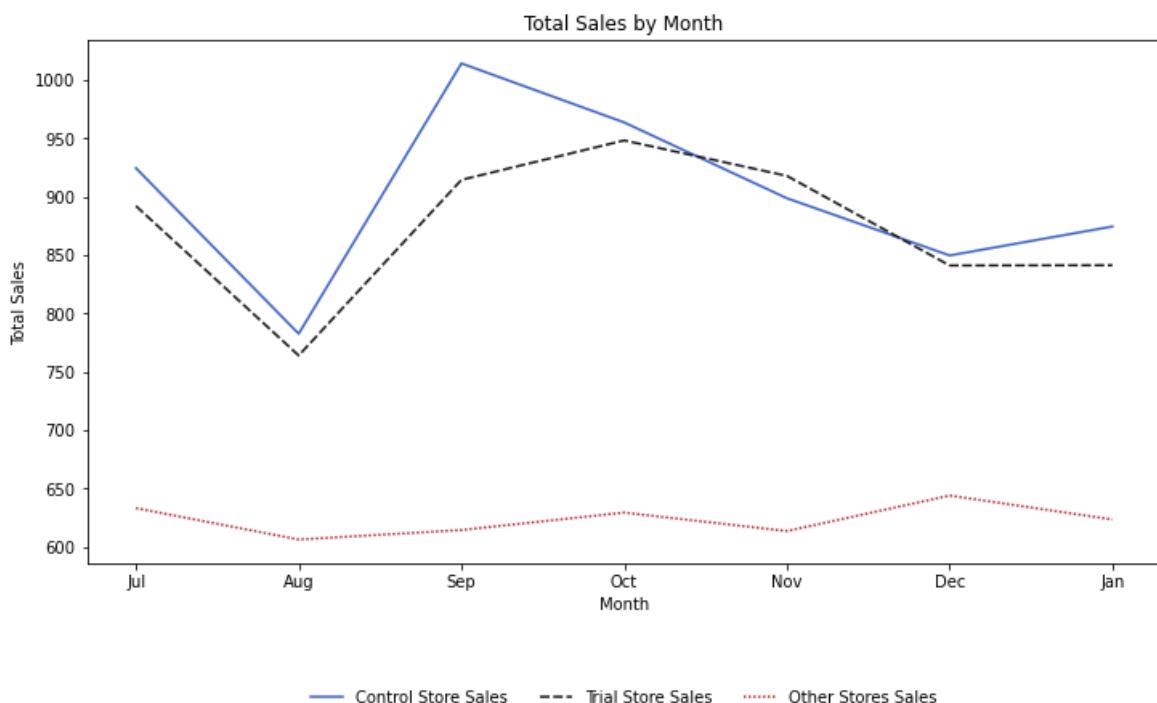
Out[77]:

Month	Control Store Sales	Trial Store Sales	Other Stores Sales
Jul	924.6	892.20	633.239922
Aug	782.7	764.05	606.326744
Sep	1014.4	914.60	614.480620
Oct	963.8	948.40	629.415504
Nov	898.8	918.00	613.593411
Dec	849.8	841.20	644.020155
Jan	874.6	841.40	623.513566

In [78]: # Plot total Sales for all 3 types of stores

```
plt.figure(figsize = (12, 6))
plt.title('Total Sales by Month')
ax = sns.lineplot(data=combinePlot, palette="icefire")

plt.ylabel("Total Sales")
sns.move_legend(
    ax, "lower center",
    bbox_to_anchor=(0.5, - .3),
    ncol=3,
    frameon=False,
)
```



## Visual checks on customer count trends by comparing the trial store to the control store and other stores.

In [79]:

```
control_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Control Store', 'no_Customers'].reset_index(inplace = True)

trial_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Trial Store', 'no_Customers'].reset_index(inplace = True)

other_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Other Stores', 'no_Customers'].reset_index(inplace = True)
other_Customer_Plot = pd.DataFrame(other_Customer_Plot.groupby('Month')['no_Customers'].sum())

# Renaming Column Names
control_Customer_Plot.rename(columns = {'no_Customers': 'Control Store Customers'}, inplace = True)
trial_Customer_Plot.rename(columns = {'no_Customers': 'Trial Store Customers'}, inplace = True)
other_Customer_Plot.rename(columns = {'no_Customers': 'Other Stores Customers'}, inplace = True)

# Concatenate
combine_Customer_Plot = pd.concat([control_Customer_Plot, trial_Customer_Plot, other_Customer_Plot], axis = 1)
```

```
combine_Customer_Plot
```

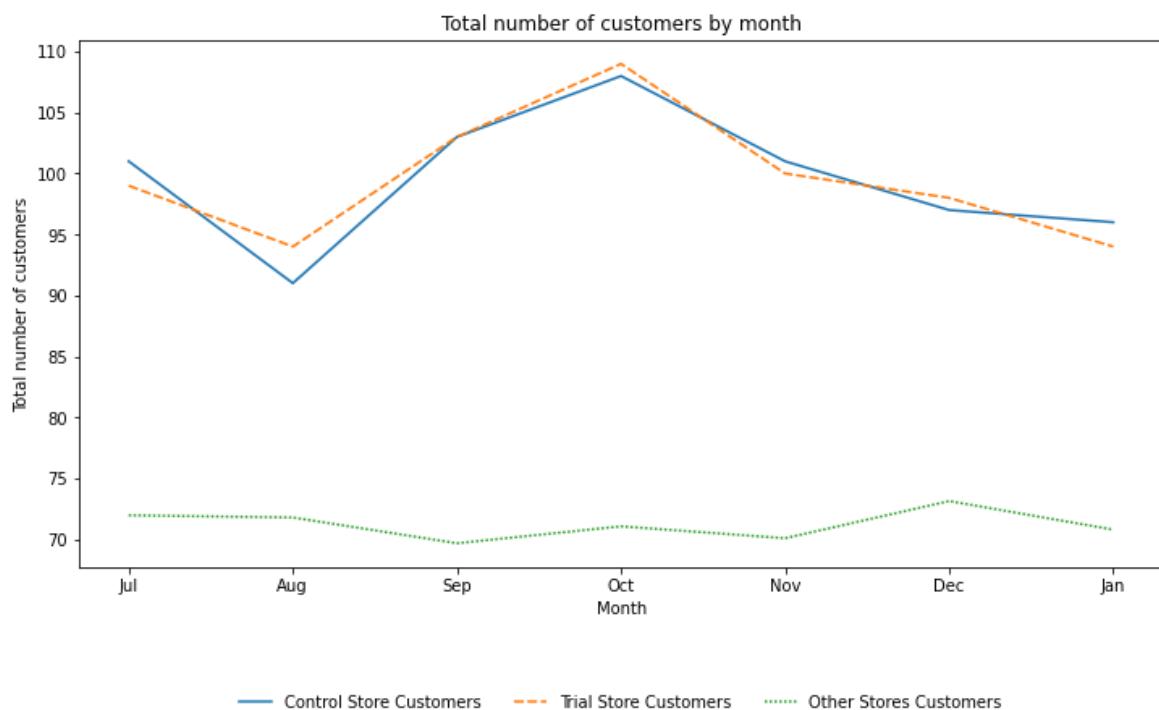
Out[79]:

	<b>Control Store Customers</b>	<b>Trial Store Customers</b>	<b>Other Stores Customers</b>
<b>Month</b>			
<b>Jul</b>	101	99	71.953488
<b>Aug</b>	91	94	71.771318
<b>Sep</b>	103	103	69.658915
<b>Oct</b>	108	109	71.046512
<b>Nov</b>	101	100	70.069767
<b>Dec</b>	97	98	73.120155
<b>Jan</b>	96	94	70.775194

In [80]: # Plot total number of customers for all 3 types of stores

```
plt.figure(figsize = (12, 6))
plt.title('Total number of customers by month')
ax1 = sns.lineplot(data=combine_Customer_Plot)

plt.ylabel("Total number of customers")
sns.move_legend(
    ax1, "lower center",
    bbox_to_anchor=(0.5, -.3),
    ncol=3,
    frameon=False,
)
```



## Assessment of trial

The trial period goes from the start of February 2019 to April 2019. We now want to see if there has been an uplift in overall chip sales.

In [81]: *# Scale pre-trial control sales to match pre-trial trial store sales*

```
trial_sales_sum = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Trial Store'].sum()
control_sales_sum = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Control Store'].sum()
scalingFactorForControlSales = trial_sales_sum / control_sales_sum
scalingFactorForControlSales
```

Out[81]: 0.9700651481287746

In [82]: *# Apply the scaling factor*

```
measureOverTimeSales = measureOverTime
scaledControlSales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] == 155]
scaledControlSales['controlSales'] = scaledControlSales['Total Sales'] * scalingFactorForControlSales
scaledControlSales
```

Out[82]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_per_trans
1793	155	201807	924.60	101	1.217822	
1794	155	201808	782.70	91	1.307692	
1795	155	201809	1014.40	103	1.398058	
1796	155	201810	963.80	108	1.259259	
1797	155	201811	898.80	101	1.316832	
1798	155	201812	849.80	97	1.237113	
1799	155	201901	874.60	96	1.302083	
1800	155	201902	891.20	95	1.315789	
1801	155	201903	804.40	94	1.255319	
1802	155	201904	844.60	99	1.212121	
1803	155	201905	922.85	106	1.283019	
1804	155	201906	857.20	95	1.273684	

Now that we have comparable sales figures for the control store, we can calculate the percentage difference between the scaled control sales and the trial store's sales during the trial period.

In [83]: *percentageDiff = scaledControlSales[['YEARMONTH', 'controlSales']]*  
*percentageDiff.reset\_index(drop = True, inplace = True)*

```
In [84]: trial_sales = measureOverTime.loc[measureOverTime['STORE_NBR'] == trial_store]
trial_sales.reset_index(drop = True, inplace = True)

percentageDiff = pd.concat([percentageDiff, trial_sales], axis = 1)
percentageDiff.rename(columns = {'Total Sales': 'Trial_Sales'}, inplace = True)
percentageDiff
```

Out[84]:

	YEARMONTH	controlSales	Trial_Sales
0	201807	896.922236	892.20
1	201808	759.269991	764.05
2	201809	984.034086	914.60
3	201810	934.948790	948.40
4	201811	871.894555	918.00
5	201812	824.361363	841.20
6	201901	848.418979	841.40
7	201902	864.522060	913.20
8	201903	780.320405	1026.80
9	201904	819.317024	848.20
10	201905	895.224622	889.30
11	201906	831.539845	838.00

```
In [85]: percentageDiff['percentageDiff'] = abs(percentageDiff.controlSales - percentageDiff.Trial_Sales)
```

Out[85]:

	YEARMONTH	controlSales	Trial_Sales	percentageDiff
0	201807	896.922236	892.20	0.005265
1	201808	759.269991	764.05	0.006296
2	201809	984.034086	914.60	0.070561
3	201810	934.948790	948.40	0.014387
4	201811	871.894555	918.00	0.052880
5	201812	824.361363	841.20	0.020426
6	201901	848.418979	841.40	0.008273
7	201902	864.522060	913.20	0.056306
8	201903	780.320405	1026.80	0.315870
9	201904	819.317024	848.20	0.035253
10	201905	895.224622	889.30	0.006618
11	201906	831.539845	838.00	0.007769

As our null hypothesis is that the trial period is the same as the pre-trial period, let's take the standard deviation based on the scaled percentage difference in the pre-trial period

```
In [86]: stdDev = stdev(percentageDiff.loc[percentageDiff['YEARMONTH'] < 201902, 'perc
stdDev
```

Out[86]: 0.025833952854772586

```
In [87]: # Note that there are 8 months in the pre-trial period , Hence 8 - 1 = 7 degr
degreesOfFreedom = 7
```

```
In [88]: measureOverTimeSales = measureOverTime
measureOverTimeSales
```

Out[88]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_pe
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3120 rows × 8 columns

In [89]: *#### Trial and control store total sales*

```
pastSales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] == tri  
pastSales.set_index('TransactionMonth', inplace = True)  
pastSales.rename(columns = {'Total Sales': 'Trial Sales'}, inplace = True)  
pastSales
```

Out[89]:

**Trial Sales**

TransactionMonth	
2018-07-01	892.20
2018-08-01	764.05
2018-09-01	914.60
2018-10-01	948.40
2018-11-01	918.00
2018-12-01	841.20
2019-01-01	841.40
2019-02-01	913.20
2019-03-01	1026.80
2019-04-01	848.20
2019-05-01	889.30
2019-06-01	838.00

In [90]: *#### Control store 95th percentile*

```
controlSales = scaledControlSales.loc[:, ['TransactionMonth', 'controlSales']]
controlSales.set_index('TransactionMonth', inplace = True)
controlSales.rename(columns = {'controlSales': 'Control Sales'}, inplace = True)
controlSales
```

Out[90]:

**Control Sales**

TransactionMonth	
2018-07-01	896.922236
2018-08-01	759.269991
2018-09-01	984.034086
2018-10-01	934.948790
2018-11-01	871.894555
2018-12-01	824.361363
2019-01-01	848.418979
2019-02-01	864.522060
2019-03-01	780.320405
2019-04-01	819.317024
2019-05-01	895.224622
2019-06-01	831.539845

```
In [91]: ##### Control store 95th percentile
controlSales['Control store 95th percentile'] = controlSales['Control Sales']

##### Control store 5th percentile

controlSales['Control store 5th percentile'] = controlSales['Control Sales']

controlSales
```

Out[91]:

TransactionMonth	Control Sales	Control store 95th percentile	Control store 5th percentile
2018-07-01	896.922236	943.264329	850.580142
2018-08-01	759.269991	798.499882	720.040101
2018-09-01	984.034086	1034.877067	933.191106
2018-10-01	934.948790	983.255636	886.641944
2018-11-01	871.894555	916.943521	826.845589
2018-12-01	824.361363	866.954388	781.768338
2019-01-01	848.418979	892.255010	804.582947
2019-02-01	864.522060	909.190104	819.854016
2019-03-01	780.320405	820.637926	740.002884
2019-04-01	819.317024	861.649419	776.984629
2019-05-01	895.224622	941.479003	848.970241
2019-06-01	831.539845	874.503767	788.575923

In [92]: trialAssessment = pd.merge(controlSales, pastSales, left\_index = True, right\_index = True)

Out[92]:

TransactionMonth	Control Sales	Control store 95th percentile	Control store 5th percentile	Trial Sales
2018-07-01	896.922236	943.264329	850.580142	892.20
2018-08-01	759.269991	798.499882	720.040101	764.05
2018-09-01	984.034086	1034.877067	933.191106	914.60
2018-10-01	934.948790	983.255636	886.641944	948.40
2018-11-01	871.894555	916.943521	826.845589	918.00
2018-12-01	824.361363	866.954388	781.768338	841.20
2019-01-01	848.418979	892.255010	804.582947	841.40
2019-02-01	864.522060	909.190104	819.854016	913.20
2019-03-01	780.320405	820.637926	740.002884	1026.80
2019-04-01	819.317024	861.649419	776.984629	848.20
2019-05-01	895.224622	941.479003	848.970241	889.30
2019-06-01	831.539845	874.503767	788.575923	838.00

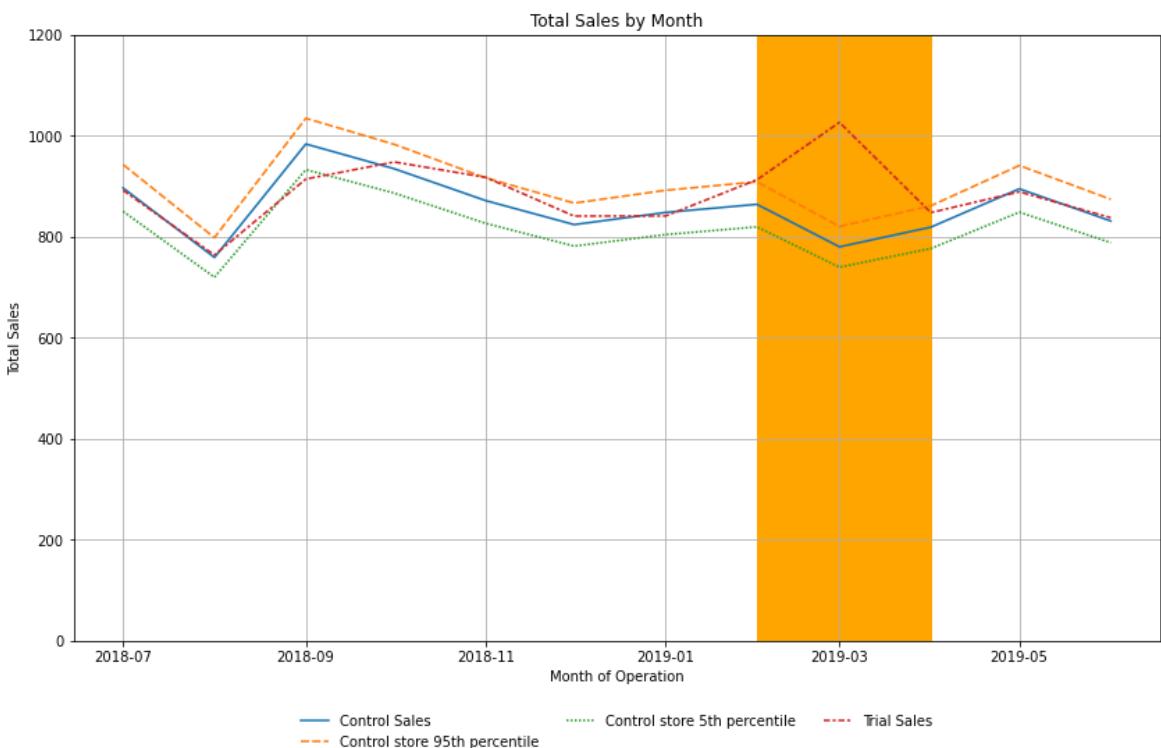
- Let's create a more visual version of this by plotting the sales of the control store, the sales of the trial stores and the 95th percentile value of sales of the control store.

```
In [93]: plt.figure(figsize = (14,8))
ax2 = sns.lineplot(data=trialAssessment)

sns.move_legend(
    ax2, "lower center",
    bbox_to_anchor=(0.5, -.2),
    ncol=3,
    frameon=False,
)

plt.title('Total Sales by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
plt.ylim((0, 1200))

plt.axvspan(*mdates.datestr2num(['2019-02-01', '2019-04-01']), color = 'orange')
plt.grid()
```



The trial period is highlighted by the orange strip.

The results show that the trial in store 86 is significantly different to its control store in the trial period as the trial store performance lies outside the 5% to 95% confidence interval of the control store in two of the three trial months.

Let's have a look at assessing this for number of customers as well.

```
In [94]: trial_customers = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Trial']
control_customers = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Control']
scalingFactorForControlCust = trial_customers / control_customers
scalingFactorForControlCust
```

Out[94]: 1.0

```
In [95]: measureOverTimeCusts = measureOverTime
measureOverTimeCusts
```

Out[95]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_per_trans
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3120 rows × 8 columns

```
In [96]: scaledControlCustomers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR'] == 272]
scaledControlCustomers
```

Out[96]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_
1793	155	201807	924.60	101	1.217822	
1794	155	201808	782.70	91	1.307692	
1795	155	201809	1014.40	103	1.398058	
1796	155	201810	963.80	108	1.259259	
1797	155	201811	898.80	101	1.316832	
1798	155	201812	849.80	97	1.237113	
1799	155	201901	874.60	96	1.302083	
1800	155	201902	891.20	95	1.315789	
1801	155	201903	804.40	94	1.255319	
1802	155	201904	844.60	99	1.212121	
1803	155	201905	922.85	106	1.283019	
1804	155	201906	857.20	95	1.273684	



In [97]: `scaledControlCustomers['Control_Customers'] = scaledControlCustomers['no_Cust  
scaledControlCustomers`

Out[97]:

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_
1793	155	201807	924.60	101	1.217822	
1794	155	201808	782.70	91	1.307692	
1795	155	201809	1014.40	103	1.398058	
1796	155	201810	963.80	108	1.259259	
1797	155	201811	898.80	101	1.316832	
1798	155	201812	849.80	97	1.237113	
1799	155	201901	874.60	96	1.302083	
1800	155	201902	891.20	95	1.315789	
1801	155	201903	804.40	94	1.255319	
1802	155	201904	844.60	99	1.212121	
1803	155	201905	922.85	106	1.283019	
1804	155	201906	857.20	95	1.273684	



In [98]: `# Calculate the percentage difference between scaled control customers. and t`

```
percentageDiff = scaledControlCustomers[['YEARMONTH', 'Control_Customers']]
percentageDiff.reset_index(drop = True, inplace = True)
percentageDiff
```

Out[98]:

	YEARMONTH	Control_Customers
0	201807	101.0
1	201808	91.0
2	201809	103.0
3	201810	108.0
4	201811	101.0
5	201812	97.0
6	201901	96.0
7	201902	95.0
8	201903	94.0
9	201904	99.0
10	201905	106.0
11	201906	95.0

In [99]:

```
trial_Customers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR']]
trial_Customers.reset_index(drop = True, inplace = True)
trial_Customers
```

Out[99]:

0	99
1	94
2	103
3	109
4	100
5	98
6	94
7	107
8	115
9	105
10	104
11	98

Name: no\_Customers, dtype: int64

In [100...]:

```
percentageDiff = pd.concat([percentageDiff, trial_Customers], axis = 1)
percentageDiff.rename(columns = {'no_Customers': 'Trial_Customers'}, inplace = True)
percentageDiff
```

Out[100...]

	YEARMONTH	Control_Customers	Trial_Customers
0	201807	101.0	99
1	201808	91.0	94
2	201809	103.0	103
3	201810	108.0	109
4	201811	101.0	100
5	201812	97.0	98
6	201901	96.0	94
7	201902	95.0	107
8	201903	94.0	115
9	201904	99.0	105
10	201905	106.0	104
11	201906	95.0	98

In [101...]

```
percentageDiff['percentageDiff'] = abs(percentageDiff.Control_Customers - per
percentageDiff
```

Out[101...]

	YEARMONTH	Control_Customers	Trial_Customers	percentageDiff
0	201807	101.0	99	0.019802
1	201808	91.0	94	0.032967
2	201809	103.0	103	0.000000
3	201810	108.0	109	0.009259
4	201811	101.0	100	0.009901
5	201812	97.0	98	0.010309
6	201901	96.0	94	0.020833
7	201902	95.0	107	0.126316
8	201903	94.0	115	0.223404
9	201904	99.0	105	0.060606
10	201905	106.0	104	0.018868
11	201906	95.0	98	0.031579

Let's again see if the difference is significant visually!

In [102...]

```
stdDev = stdev(percentageDiff.loc[percentageDiff['YEARMONTH'] < 201902, 'perc
degreesOfFreedom = 7
```

As our null hypothesis is that the trial period is the same as the pre-trial period, let's take the standard deviation based on the scaledpercentage difference in the pre-trial period

```
In [103...]: percentageDiff['T_value'] = (percentageDiff['percentageDiff'] - 0) / stdDev
percentageDiff.loc[(percentageDiff['YEARMONTH'] > 201901) & (percentageDiff['
```

```
Out[103...]: 7    11.819082
             8    20.903430
             9    5.670772
Name: T_value, dtype: float64
```

```
In [104...]: ##### 95th percentile of the t distribution with degreesOfFreedom = 7

t95 = t.isf(0.05, degreesOfFreedom)
print('95th percentile of the t distribution with the degrees Of Freedom 7')
```

```
95th percentile of the t distribution with the degrees Of Freedom 7      - 1.8945
786050613054
```

We Can see that the t-value is much greater than the 95th percentile of the t distribution

```
In [105...]: controlCustomers = scaledControlCustomers.loc[:, ['TransactionMonth', 'Control']]
controlCustomers.set_index('TransactionMonth', inplace = True)
controlCustomers
```

Control_Customers	
TransactionMonth	
2018-07-01	101.0
2018-08-01	91.0
2018-09-01	103.0
2018-10-01	108.0
2018-11-01	101.0
2018-12-01	97.0
2019-01-01	96.0
2019-02-01	95.0
2019-03-01	94.0
2019-04-01	99.0
2019-05-01	106.0
2019-06-01	95.0

```
In [106...]: trialCustomers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR'] == 1]
trialCustomers.set_index('TransactionMonth', inplace = True)
```

```
trialCustomers.rename(columns = {'no_Customers': 'Trial Customers'}, inplace=True)
```

Out[106...]

**Trial Customers**

TransactionMonth	
2018-07-01	99
2018-08-01	94
2018-09-01	103
2018-10-01	109
2018-11-01	100
2018-12-01	98
2019-01-01	94
2019-02-01	107
2019-03-01	115
2019-04-01	105
2019-05-01	104
2019-06-01	98

In [107...]

```
#### Control store 95th percentile
controlCustomers['Control 95th % confidence Interval'] = controlCustomers['Control 95th % confidence Interval'].round(0)
controlCustomers['Control 5th % confidence Interval'] = controlCustomers['Control 5th % confidence Interval'].round(0)

# Merge trial store as well to the control store dataframe.
trial_86_Customer_Assessment = pd.merge(controlCustomers, trialCustomers, left_index=True, right_index=True)
```

In [108...]

```
trial_86_Customer_Assessment
```

Out[108...]

Control_Customers	Control 95th % confidence Interval	Control 5th % confidence Interval	Trial Customers
TransactionMonth			
2018-07-01	101.0	103.158864	98.841136
2018-08-01	91.0	92.945115	89.054885
2018-09-01	103.0	105.201614	100.798386
2018-10-01	108.0	110.308488	105.691512
2018-11-01	101.0	103.158864	100
2018-12-01	97.0	99.073364	94.926636
2019-01-01	96.0	98.051989	93.948011
2019-02-01	95.0	97.030614	92.969386
2019-03-01	94.0	96.009240	91.990760
2019-04-01	99.0	101.116114	96.883886
2019-05-01	106.0	108.265738	103.734262
2019-06-01	95.0	97.030614	92.969386

In [109...]

```

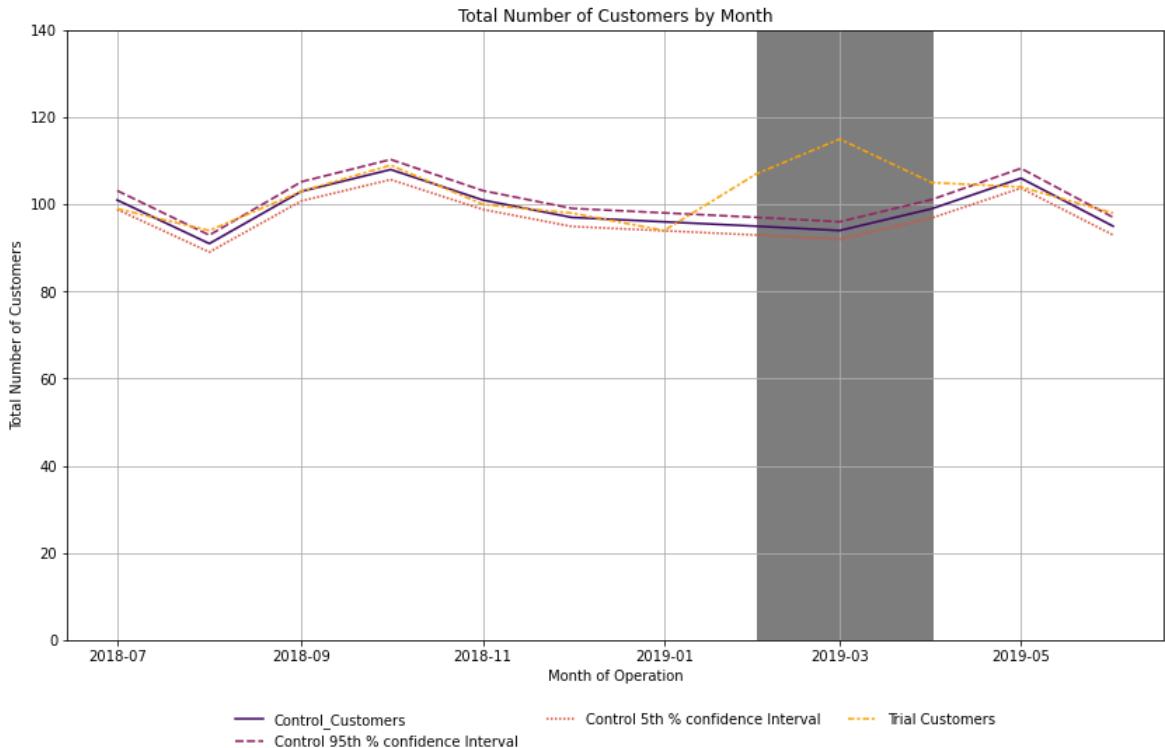
plt.figure(figsize = (14,8))
ax4 = sns.lineplot(data=trial_86_Customer_Assessment, palette = 'inferno')

sns.move_legend(
    ax4, "lower center",
    bbox_to_anchor=(0.5, -.2),
    ncol=3,
    frameon=False,
)

plt.title('Total Number of Customers by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Number of Customers')
plt.ylim((0,140))

plt.axvspan(*mdates.datestr2num(['2019-02-01', '2019-04-01']), color = 'grey')
plt.grid()

```



The trial period is highlighted by the grey strip.

The number of clients appears to have increased dramatically over the course of the three months. This would appear to indicate that the trial greatly increased the number of patrons in trial shop 86, but as we observed, sales were not significantly higher.

We should check with the Category Manager if there were special deals in the trial store that may have resulted in lower prices, impacting the results.

## Trial store 88

```
In [110...]: trial_store = 88

corr_Sales = calculateCorrelation(preTrialMeasures, 'Total Sales', trial_stor
corr_Customers = calculateCorrelation(preTrialMeasures, 'no_Customers', trial

# Compute magnitude with trial store 88
magnitude_Sales = calculateMagnitudeDistance(preTrialMeasures, 'Total Sales',
magnitude_Customers = calculateMagnitudeDistance(preTrialMeasures, 'no_Custom

In [111...]: score_Sales = pd.concat([corr_Sales, magnitude_Sales['Magnitude']], axis = 1)
score_Sales
```

Out[111...]

	<b>Store1</b>	<b>Store2</b>	<b>corr_measure</b>	<b>Magnitude</b>
<b>0</b>	88.0	1.0	0.813636	0.548959
<b>1</b>	88.0	2.0	-0.067927	0.541212
<b>2</b>	88.0	3.0	-0.507847	0.458109
<b>3</b>	88.0	4.0	-0.745566	0.484447
<b>4</b>	88.0	5.0	0.190330	0.496409
...	...	...	...	...
<b>255</b>	88.0	268.0	-0.021429	0.522847
<b>256</b>	88.0	269.0	-0.172578	0.564856
<b>257</b>	88.0	270.0	-0.723272	0.513329
<b>258</b>	88.0	271.0	-0.103037	0.429220
<b>259</b>	88.0	272.0	-0.772772	0.617809

260 rows × 4 columns

In [112...]: `score_Customers = pd.concat([corr_Customers, magnitude_Customers['Magnitude']]`In [113...]: `# Add an additional column which calculates the weighted average`

```
corr_weight = 0.5
score_Sales['scoreSales'] = corr_weight * score_Sales['corr_measure'] + (1 - score_Customers['scoreCustomers']) = corr_weight * score_Customers['corr_measu
```

In [114...]: `score_Sales.set_index(['Store1', 'Store2'], inplace = True)
score_Customers.set_index(['Store1', 'Store2'], inplace = True)`

```
score_Control = pd.concat([score_Sales['scoreSales'], score_Customers['scoreC
score_Control
```

Out[114...]

	scoreSales	scoreCustomers
Store1	Store2	
<b>88.0</b>	<b>1.0</b>	0.681297
	<b>2.0</b>	0.236643
	<b>3.0</b>	-0.024869
	<b>4.0</b>	-0.130559
	<b>5.0</b>	0.343370
	...	...
<b>268.0</b>		0.250709
<b>269.0</b>		0.196139
<b>270.0</b>		-0.104971
<b>271.0</b>		0.163091
<b>272.0</b>		-0.077482

260 rows × 2 columns

In [115...]

```
# Add a new column 'finalControlScore' to 'score_Control'.
score_Control['finalControlScore'] = 0.5 * (score_Control['scoreSales'] + score_Control['scoreCustomers'])
```

Out[115...]

	scoreSales	scoreCustomers	finalControlScore
Store1	Store2		
<b>88.0</b>	<b>1.0</b>	0.681297	0.506268
	<b>2.0</b>	0.236643	-0.083332
	<b>3.0</b>	-0.024869	0.289205
	<b>4.0</b>	-0.130559	-0.011175
	<b>5.0</b>	0.343370	0.304965
	...	...	...
<b>268.0</b>		0.250709	0.451086
<b>269.0</b>		0.196139	0.119600
<b>270.0</b>		-0.104971	0.013748
<b>271.0</b>		0.163091	0.240799
<b>272.0</b>		-0.077482	0.080231

260 rows × 3 columns

```
In [116]: score_Control.sort_values(by = 'finalControlScore', ascending = False).head()
```

```
Out[116]:
```

	scoreSales	scoreCustomers	finalControlScore
Store1	Store2		
88.0	178.0	0.650803	0.707828
	14.0	0.646064	0.685774
	134.0	0.775084	0.540154
	237.0	0.451974	0.777235
	187.0	0.616752	0.594560
			0.605656

We've now found store 237 to be a suitable control store for trial store 88.

Now that we have found a control store, let's check visually if the drivers are indeed similar in the period before the trial.

```
In [117]: control_store = 237

pastSales = preTrialMeasures

store_type = []

for i in pastSales['STORE_NBR']:
    if i == trial_store:
        store_type.append('Trial Store')
    elif i == control_store:
        store_type.append('Control Store')
    else:
        store_type.append('Other Stores')

pastSales['store_type'] = store_type
pastSales.head()
```

Out[117...]

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_per_cu
0	1	201807	206.9	49	1.061224	1
1	1	201808	176.1	42	1.023810	1
2	1	201809	278.8	59	1.050847	1
3	1	201810	188.1	44	1.022727	1
4	1	201811	192.6	46	1.021739	1



In [118...]

```
controlPlot = pastSales.loc[pastSales['store_type'] == 'Control Store', ['Month']]
controlPlot.set_index('Month', inplace = True)

trialPlot = pastSales.loc[pastSales['store_type'] == 'Trial Store', ['Month']]
trialPlot.set_index('Month', inplace = True)

otherPlot = pastSales.loc[pastSales['store_type'] == 'Other Stores', ['Month']]
otherPlot = pd.DataFrame(otherPlot.groupby('Month')['Total Sales'].mean())

# Renaming Column Names
controlPlot.rename(columns = {'Total Sales': 'Control Store Sales'}, inplace = True)
trialPlot.rename(columns = {'Total Sales': 'Trial Store Sales'}, inplace = True)
otherPlot.rename(columns = {'Total Sales': 'Other Stores Sales'}, inplace = True)

# Concatenate
combinePlot = pd.concat([controlPlot, trialPlot, otherPlot], axis = 1)

combinePlot
```

Out[118...]

Month	Control Store Sales	Trial Store Sales	Other Stores Sales
Jul	1448.4	1310.0	629.590310
Aug	1367.8	1323.8	601.889341
Sep	1322.2	1423.0	611.317054
Oct	1348.3	1352.4	626.359302
Nov	1397.6	1382.8	609.858527
Dec	1265.0	1325.2	640.534884
Jan	1219.7	1266.4	620.528682

Month	Control Store Sales	Trial Store Sales	Other Stores Sales
Jul	1448.4	1310.0	629.590310
Aug	1367.8	1323.8	601.889341
Sep	1322.2	1423.0	611.317054
Oct	1348.3	1352.4	626.359302
Nov	1397.6	1382.8	609.858527
Dec	1265.0	1325.2	640.534884
Jan	1219.7	1266.4	620.528682

In [119...]

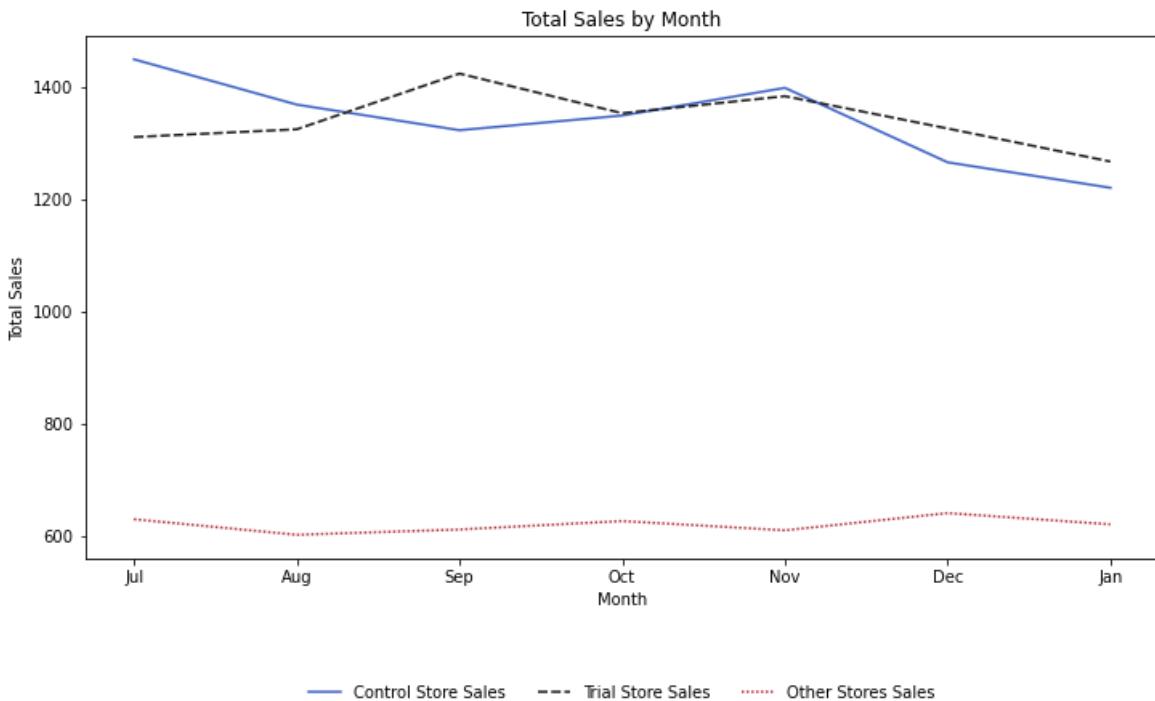
```
# Plot total Sales for all 3 types of stores
```

```

plt.figure(figsize = (12, 6))
plt.title('Total Sales by Month')
ax5 = sns.lineplot(data=combinePlot, palette="icefire")

plt.ylabel("Total Sales")
sns.move_legend(
    ax5, "lower center",
    bbox_to_anchor=(0.5, - .3),
    ncol=3,
    frameon=False,
)
)

```



## Visual checks on customer count trends by comparing the trial store to the control store and other stores.

```

In [120...]: control_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Control Store']
control_Customer_Plot.set_index('Month', inplace = True)

trial_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Trial Store',
trial_Customer_Plot.set_index('Month', inplace = True)

other_Customer_Plot = pastSales.loc[pastSales['store_type'] == 'Other Stores']
other_Customer_Plot = pd.DataFrame(other_Customer_Plot.groupby('Month')['no_Cus
tomers'].sum())

# Renaming Column Names
control_Customer_Plot.rename(columns = {'no_Customers': 'Control Store Customers'}, i
nplace = True)
trial_Customer_Plot.rename(columns = {'no_Customers': 'Trial Store Customers'}, inplace = T
rue)
other_Customer_Plot.rename(columns = {'no_Customers': 'Other Stores Customers'}, inplace = T
rue)

# Concatenate
combine_Customer_Plot = pd.concat([control_Customer_Plot, trial_Customer_Plot, other_Cus
tomer_Plot])

combine_Customer_Plot

```

Out[120...]

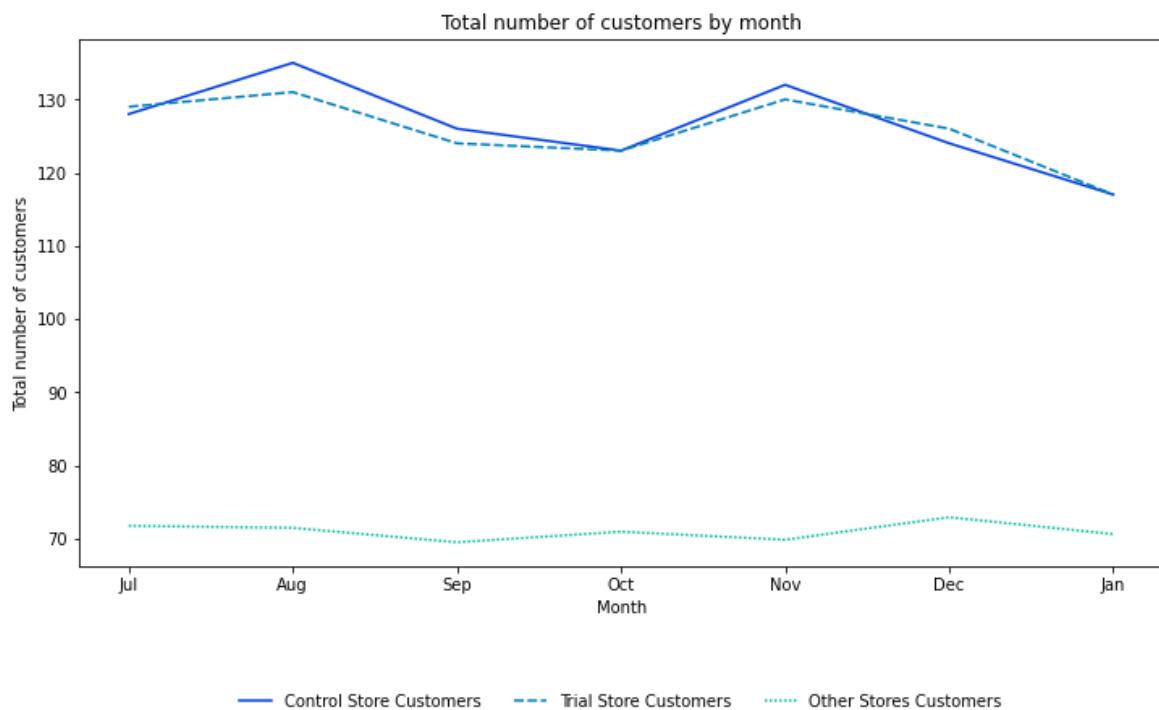
	Control Store Customers	Trial Store Customers	Other Stores Customers
<b>Month</b>			
<b>Jul</b>	128	129	71.732558
<b>Aug</b>	135	131	71.457364
<b>Sep</b>	126	124	69.488372
<b>Oct</b>	123	123	70.934109
<b>Nov</b>	132	130	69.833333
<b>Dec</b>	124	126	72.906977
<b>Jan</b>	117	117	70.604651

In [121...]

```
# Plot total number of customers for all 3 types of stores
```

```
plt.figure(figsize = (12, 6))
plt.title('Total number of customers by month')
ax6 = sns.lineplot(data=combine_Customer_Plot, palette="winter")

plt.ylabel("Total number of customers")
sns.move_legend(
    ax6, "lower center",
    bbox_to_anchor=(0.5, - .3),
    ncol=3,
    frameon=False,
)
```



## Assessment of trial

The trial period goes from the start of February 2019 to April 2019. We now want to see if there has been an uplift in overall chip sales.

In [122...]

```
# Scale pre-trial control sales to match pre-trial trial store sales

trial_sales_sum = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Tri
control_sales_sum = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'C
scalingFactorForControlSales = trial_sales_sum / control_sales_sum
scalingFactorForControlSales
```

Out[122...]

1.0015583306649594

In [123...]

```
# Apply the scaling factor

measureOverTimeSales = measureOverTime
scaledControlSales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR
scaledControlSales['controlSales'] = scaledControlSales['Total Sales'] * scal
scaledControlSales
```

Out[123...]

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_p
2747	237	201807	1448.4	128	1.265625	
2748	237	201808	1367.8	135	1.222222	
2749	237	201809	1322.2	126	1.182540	
2750	237	201810	1348.3	123	1.195122	
2751	237	201811	1397.6	132	1.219697	
2752	237	201812	1265.0	124	1.161290	
2753	237	201901	1219.7	117	1.188034	
2754	237	201902	1404.8	126	1.246032	
2755	237	201903	1208.2	119	1.126050	
2756	237	201904	1204.6	120	1.125000	
2757	237	201905	1199.3	129	1.155039	
2758	237	201906	1153.6	119	1.100840	

Now that we have comparable sales figures for the control store, we can calculate the percentage difference between the scaled control sales and the trial store's sales during the trial period.

In [124...]

```
percentageDiff = scaledControlSales[['YEARMONTH', 'controlSales']]
percentageDiff.reset_index(drop = True, inplace = True)
```

```
In [125...]: trial_sales = measureOverTime.loc[measureOverTime['STORE_NBR'] == trial_store]
trial_sales.reset_index(drop = True, inplace = True)

percentageDiff = pd.concat([percentageDiff, trial_sales], axis = 1)
percentageDiff.rename(columns = {'Total Sales': 'Trial_Sales'}, inplace = True)
percentageDiff
```

Out[125...]:

	YEARMONTH	controlSales	Trial_Sales
0	201807	1450.657086	1310.00
1	201808	1369.931485	1323.80
2	201809	1324.260425	1423.00
3	201810	1350.401097	1352.40
4	201811	1399.777923	1382.80
5	201812	1266.971288	1325.20
6	201901	1221.600696	1266.40
7	201902	1406.989143	1370.20
8	201903	1210.082775	1477.20
9	201904	1206.477165	1439.40
10	201905	1201.168906	1308.25
11	201906	1155.397690	1354.60

In [126...]:

```
percentageDiff['percentageDiff'] = abs(percentageDiff.controlSales - percentageDiff.Trial_Sales)
```

Out[126...]

	YEARMONTH	controlSales	Trial_Sales	percentageDiff
0	201807	1450.657086	1310.00	0.096961
1	201808	1369.931485	1323.80	0.033674
2	201809	1324.260425	1423.00	0.074562
3	201810	1350.401097	1352.40	0.001480
4	201811	1399.777923	1382.80	0.012129
5	201812	1266.971288	1325.20	0.045959
6	201901	1221.600696	1266.40	0.036673
7	201902	1406.989143	1370.20	0.026147
8	201903	1210.082775	1477.20	0.220743
9	201904	1206.477165	1439.40	0.193060
10	201905	1201.168906	1308.25	0.089147
11	201906	1155.397690	1354.60	0.172410

As our null hypothesis is that the trial period is the same as the pre-trial period, let's take the standard deviation based on the scaled percentage difference in the pre-trial period

In [127...]

```
stdDev = stdev(percentageDiff.loc[percentageDiff['YEARMONTH'] < 201902, 'perc
stdDev
```

Out[127...]

```
0.03346786730307888
```

In [128...]

```
# Note that there are 8 months in the pre-trial period , Hence 8 - 1 = 7 degr
degreesOfFreedom = 7
```

In [129...]

```
#### We will test with a null hypothesis of there being 0 difference between
percentageDiff['tValue'] = (percentageDiff['percentageDiff'] - 0) / stdDev
percentageDiff.loc[(percentageDiff['YEARMONTH'] < 201905 ) & (percentageDiff[
```

Out[129...]

```
7    0.781270
8    6.595668
9    5.768527
Name: tValue, dtype: float64
```

In [130...]

```
#### Find the 95th percentile of the t distribution with the degreesOfFreedom
t95 = t.isf(0.05, degreesOfFreedom)
print('95th percentile of the t distribution with the degrees Of Freedom 7
```

```
95th percentile of the t distribution with the degrees Of Freedom 7 - 1.8945
786050613054
```

- We can observe that the t-value is much larger than the 95th percentile value of the t-distribution for March and April - i.e. the increase in sales in the trial store in March and April is statistically greater than in the control store.

In [131...]

```
measureOverTimeSales = measureOverTime
measureOverTimeSales
```

Out[131...]

	<b>STORE_NBR</b>	<b>YEARMONTH</b>	<b>Total Sales</b>	<b>no_Customers</b>	<b>trans_per_customer</b>	<b>chips_pe</b>
<b>0</b>	1	201807	206.9	49	1.061224	
<b>1</b>	1	201808	176.1	42	1.023810	
<b>2</b>	1	201809	278.8	59	1.050847	
<b>3</b>	1	201810	188.1	44	1.022727	
<b>4</b>	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
<b>3164</b>	272	201902	395.5	45	1.066667	
<b>3165</b>	272	201903	442.3	50	1.060000	
<b>3166</b>	272	201904	445.1	54	1.018519	
<b>3167</b>	272	201905	314.6	34	1.176471	
<b>3168</b>	272	201906	312.1	34	1.088235	

3120 rows × 8 columns



In [132...]

#### Trial and control store total sales

```
pastSales = measureOverTimeSales.loc[measureOverTimeSales['STORE_NBR'] == tri
pastSales.set_index('TransactionMonth', inplace = True)
pastSales.rename(columns = {'Total Sales': 'Trial Sales'}, inplace = True)
pastSales
```

Out[132...]

**Trial Sales**

TransactionMonth	
2018-07-01	1310.00
2018-08-01	1323.80
2018-09-01	1423.00
2018-10-01	1352.40
2018-11-01	1382.80
2018-12-01	1325.20
2019-01-01	1266.40
2019-02-01	1370.20
2019-03-01	1477.20
2019-04-01	1439.40
2019-05-01	1308.25
2019-06-01	1354.60

In [133...]

```
#### Control store 95th percentile

controlSales = scaledControlSales.loc[:, ['TransactionMonth', 'controlSales']]
controlSales.set_index('TransactionMonth', inplace = True)
controlSales.rename(columns = {'controlSales': 'Control Sales'}, inplace = True)
controlSales
```

Out[133...]

**Control Sales**

TransactionMonth	
2018-07-01	1450.657086
2018-08-01	1369.931485
2018-09-01	1324.260425
2018-10-01	1350.401097
2018-11-01	1399.777923
2018-12-01	1266.971288
2019-01-01	1221.600696
2019-02-01	1406.989143
2019-03-01	1210.082775
2019-04-01	1206.477165
2019-05-01	1201.168906
2019-06-01	1155.397690

In [134...]

```
#### Control store 95th percentile
controlSales['Control store 95th percentile'] = controlSales['Control Sales']

#### Control store 5th percentile
controlSales['Control store 5th percentile'] = controlSales['Control Sales']

controlSales
```

Out[134...]

TransactionMonth	Control Sales	Control store 95th percentile	Control store 5th percentile
2018-07-01	1450.657086	1547.757884	1353.556288
2018-08-01	1369.931485	1461.628855	1278.234114
2018-09-01	1324.260425	1412.900769	1235.620080
2018-10-01	1350.401097	1440.791187	1260.011008
2018-11-01	1399.777923	1493.473086	1306.082759
2018-12-01	1266.971288	1351.776942	1182.165634
2019-01-01	1221.600696	1303.369436	1139.831956
2019-02-01	1406.989143	1501.166995	1312.811291
2019-03-01	1210.082775	1291.080555	1129.084996
2019-04-01	1206.477165	1287.233600	1125.720730
2019-05-01	1201.168906	1281.570029	1120.767783
2019-06-01	1155.397690	1232.735083	1078.060297

In [135...]

```
trialAssessment = pd.merge(controlSales, pastSales, left_index = True, right_index = True)
trialAssessment
```

Out[135...]

TransactionMonth	Control Sales	Control store 95th percentile	Control store 5th percentile	Trial Sales
2018-07-01	1450.657086	1547.757884	1353.556288	1310.00
2018-08-01	1369.931485	1461.628855	1278.234114	1323.80
2018-09-01	1324.260425	1412.900769	1235.620080	1423.00
2018-10-01	1350.401097	1440.791187	1260.011008	1352.40
2018-11-01	1399.777923	1493.473086	1306.082759	1382.80
2018-12-01	1266.971288	1351.776942	1182.165634	1325.20
2019-01-01	1221.600696	1303.369436	1139.831956	1266.40
2019-02-01	1406.989143	1501.166995	1312.811291	1370.20
2019-03-01	1210.082775	1291.080555	1129.084996	1477.20
2019-04-01	1206.477165	1287.233600	1125.720730	1439.40
2019-05-01	1201.168906	1281.570029	1120.767783	1308.25
2019-06-01	1155.397690	1232.735083	1078.060297	1354.60

- Let's create a more visual version of this by plotting the sales of the control store, the sales of the trial stores and the 95th percentile value of sales of the control store.

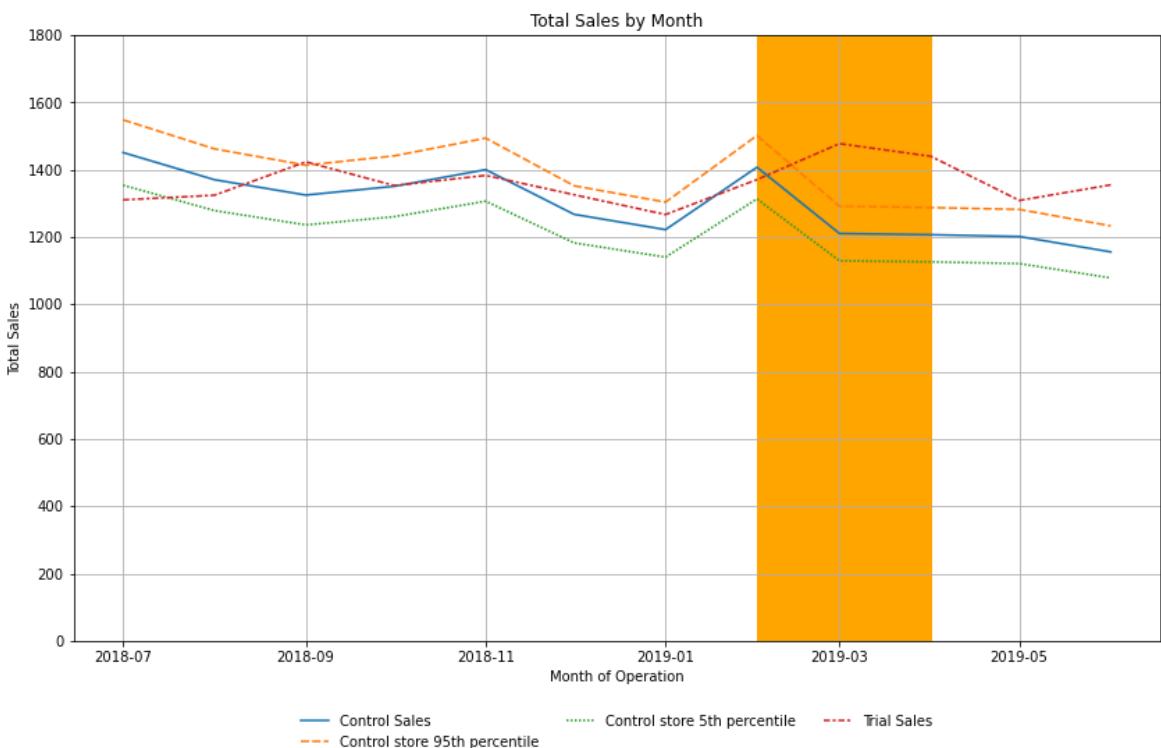
In [136...]

```
plt.figure(figsize = (14,8))
ax7 = sns.lineplot(data=trialAssessment)

sns.move_legend(
    ax7, "lower center",
    bbox_to_anchor=(0.5, -.2),
    ncol=3,
    frameon=False,
)

plt.title('Total Sales by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Sales')
plt.ylim((0, 1800))

plt.axvspan(*mdates.datestr2num(['2019-02-01', '2019-04-01']), color = 'orange')
plt.grid()
```



The trial period is highlighted by the orange strip.

The results show that the trial in store 88 is significantly different to its control store in the trial period as the trial store performance lies outside the 5% to 95% confidence interval of the control store in two of the three trial months.

Let's have a look at assessing this for number of customers as well.

```
In [137...]: trial_customers = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Trial']
control_customers = preTrialMeasures.loc[preTrialMeasures['store_type'] == 'Control']
scalingFactorForControlCust = trial_customers / control_customers
scalingFactorForControlCust
```

```
Out[137...]: 0.9943502824858758
```

```
In [138...]: measureOverTimeCusts = measureOverTime
measureOverTimeCusts
```

```
Out[138...]:
```

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_per_trans
0	1	201807	206.9	49	1.061224	
1	1	201808	176.1	42	1.023810	
2	1	201809	278.8	59	1.050847	
3	1	201810	188.1	44	1.022727	
4	1	201811	192.6	46	1.021739	
...	...	...	...	...	...	...
3164	272	201902	395.5	45	1.066667	
3165	272	201903	442.3	50	1.060000	
3166	272	201904	445.1	54	1.018519	
3167	272	201905	314.6	34	1.176471	
3168	272	201906	312.1	34	1.088235	

3120 rows × 8 columns



```
In [139...]: scaledControlCustomers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR'] == 272]
scaledControlCustomers
```

Out[139...]

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_p
2747	237	201807	1448.4	128	1.265625	
2748	237	201808	1367.8	135	1.222222	
2749	237	201809	1322.2	126	1.182540	
2750	237	201810	1348.3	123	1.195122	
2751	237	201811	1397.6	132	1.219697	
2752	237	201812	1265.0	124	1.161290	
2753	237	201901	1219.7	117	1.188034	
2754	237	201902	1404.8	126	1.246032	
2755	237	201903	1208.2	119	1.126050	
2756	237	201904	1204.6	120	1.125000	
2757	237	201905	1199.3	129	1.155039	
2758	237	201906	1153.6	119	1.100840	



In [140...]

```
scaledControlCustomers['Control_Customers'] = scaledControlCustomers['no_Customers']
scaledControlCustomers
```

Out[140...]

	STORE_NBR	YEARMONTH	Total Sales	no_Customers	trans_per_customer	chips_p
2747	237	201807	1448.4	128	1.265625	
2748	237	201808	1367.8	135	1.222222	
2749	237	201809	1322.2	126	1.182540	
2750	237	201810	1348.3	123	1.195122	
2751	237	201811	1397.6	132	1.219697	
2752	237	201812	1265.0	124	1.161290	
2753	237	201901	1219.7	117	1.188034	
2754	237	201902	1404.8	126	1.246032	
2755	237	201903	1208.2	119	1.126050	
2756	237	201904	1204.6	120	1.125000	
2757	237	201905	1199.3	129	1.155039	
2758	237	201906	1153.6	119	1.100840	



In [141...]

```
# Calculate the percentage difference between scaled control customers. and t
```

```
percentageDiff = scaledControlCustomers[['YEARMONTH', 'Control_Customers']]
percentageDiff.reset_index(drop = True, inplace = True)
percentageDiff
```

Out[141...]

	YEARMONTH	Control_Customers
0	201807	127.276836
1	201808	134.237288
2	201809	125.288136
3	201810	122.305085
4	201811	131.254237
5	201812	123.299435
6	201901	116.338983
7	201902	125.288136
8	201903	118.327684
9	201904	119.322034
10	201905	128.271186
11	201906	118.327684

In [142...]

```
trial_Customers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR']]
trial_Customers.reset_index(drop = True, inplace = True)
trial_Customers
```

Out[142...]

0	129
1	131
2	124
3	123
4	130
5	126
6	117
7	124
8	134
9	128
10	128
11	121

Name: no\_Customers, dtype: int64

In [143...]

```
percentageDiff = pd.concat([percentageDiff, trial_Customers], axis = 1)
percentageDiff.rename(columns = {'no_Customers': 'Trial_Customers'}, inplace = True)
percentageDiff
```

Out[143...]

	YEARMONTH	Control_Customers	Trial_Customers
0	201807	127.276836	129
1	201808	134.237288	131
2	201809	125.288136	124
3	201810	122.305085	123
4	201811	131.254237	130
5	201812	123.299435	126
6	201901	116.338983	117
7	201902	125.288136	124
8	201903	118.327684	134
9	201904	119.322034	128
10	201905	128.271186	128
11	201906	118.327684	121

In [144...]

```
percentageDiff['percentageDiff'] = abs(percentageDiff.Control_Customers - per
percentageDiff
```

Out[144...]

	YEARMONTH	Control_Customers	Trial_Customers	percentageDiff
0	201807	127.276836	129	0.013539
1	201808	134.237288	131	0.024116
2	201809	125.288136	124	0.010281
3	201810	122.305085	123	0.005682
4	201811	131.254237	130	0.009556
5	201812	123.299435	126	0.021902
6	201901	116.338983	117	0.005682
7	201902	125.288136	124	0.010281
8	201903	118.327684	134	0.132448
9	201904	119.322034	128	0.072727
10	201905	128.271186	128	0.002114
11	201906	118.327684	121	0.022584

Let's again see if the difference is significant visually!

In [145...]

```
stdDev = stdev(percentageDiff.loc[percentageDiff['YEARMONTH'] < 201902, 'perc
degreesOfFreedom = 7
```

As our null hypothesis is that the trial period is the same as the pre-trial period, let's take the standard deviation based on the scaledpercentage difference in the pre-trial period

```
In [146...]: percentageDiff['T_value'] = (percentageDiff['percentageDiff'] - 0) / stdDev
percentageDiff.loc[(percentageDiff['YEARMONTH'] > 201901) & (percentageDiff['
```

```
Out[146...]: 7    1.387456
             8    17.873693
             9    9.814423
Name: T_value, dtype: float64
```

```
In [147...]: ##### Find the 95th percentile of the t distribution with the degreesOfFreedom

t95 = t.isf(0.05, degreesOfFreedom)
print('95th percentile of the t distribution with the degrees Of Freedom 7')

95th percentile of the t distribution with the degrees Of Freedom 7      - 1.8945
786050613054
```

We Can see that the t-value is much greater than the 95th percentile of the t distribution

```
In [148...]: controlCustomers = scaledControlCustomers.loc[:, ['TransactionMonth', 'Control']]
controlCustomers.set_index('TransactionMonth', inplace = True)
controlCustomers
```

Control_Customers	
TransactionMonth	
2018-07-01	127.276836
2018-08-01	134.237288
2018-09-01	125.288136
2018-10-01	122.305085
2018-11-01	131.254237
2018-12-01	123.299435
2019-01-01	116.338983
2019-02-01	125.288136
2019-03-01	118.327684
2019-04-01	119.322034
2019-05-01	128.271186
2019-06-01	118.327684

```
In [149...]: trialCustomers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR'] =
trialCustomers.set_index('TransactionMonth', inplace = True)
```

```
trialCustomers.rename(columns = {'no_Customers': 'Trial Customers'}, inplace = True)  
trialCustomers
```

Out[149...]

**Trial Customers**

TransactionMonth	
2018-07-01	129
2018-08-01	131
2018-09-01	124
2018-10-01	123
2018-11-01	130
2018-12-01	126
2019-01-01	117
2019-02-01	124
2019-03-01	134
2019-04-01	128
2019-05-01	128
2019-06-01	121

In [150...]

```
trialCustomers = measureOverTimeCusts.loc[measureOverTimeCusts['STORE_NBR'] == 1]  
trialCustomers.set_index('TransactionMonth', inplace = True)  
trialCustomers.rename(columns = {'no_Customers': 'Trial Customers'}, inplace = True)  
trialCustomers
```

Out[150...]

**Trial Customers**

TransactionMonth	
2018-07-01	129
2018-08-01	131
2018-09-01	124
2018-10-01	123
2018-11-01	130
2018-12-01	126
2019-01-01	117
2019-02-01	124
2019-03-01	134
2019-04-01	128
2019-05-01	128
2019-06-01	121

In [151...]

```
#### Control store 95th percentile
controlCustomers['Control 95th % confidence Interval'] = controlCustomers['Co
controlCustomers['Control 5th % confidence Interval'] = controlCustomers['Con

# Merge trial store as well to the control store dataframe.
trial_88_Customer_Assessment = pd.merge(controlCustomers, trialCustomers, lef
```

In [152...]

```
trial_88_Customer_Assessment
```

Out[152...]

Control_Customers	Control 95th % confidence Interval	Control 5th % confidence Interval	Trial Customers
TransactionMonth			
2018-07-01	127.276836	129.163141	125.390531
2018-08-01	134.237288	136.226750	132.247826
2018-09-01	125.288136	127.144967	123.431304
2018-10-01	122.305085	124.117706	120.492464
2018-11-01	131.254237	133.199489	129.308985
2018-12-01	123.299435	125.126793	121.472077
2019-01-01	116.338983	118.063184	114.614782
2019-02-01	125.288136	127.144967	123.431304
2019-03-01	118.327684	120.081358	116.574010
2019-04-01	119.322034	121.090445	117.553623
2019-05-01	128.271186	130.172228	126.370145
2019-06-01	118.327684	120.081358	116.574010

In [153...]

```

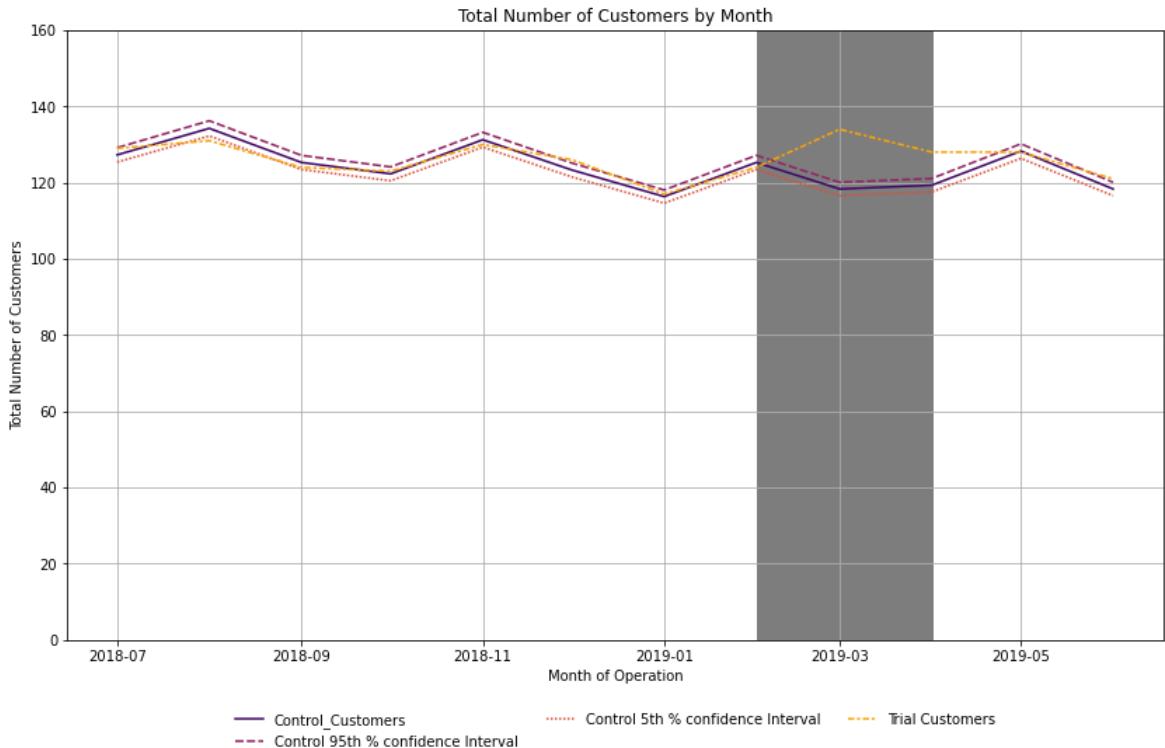
plt.figure(figsize = (14,8))
ax8 = sns.lineplot(data=trial_88_Customer_Assessment, palette = 'inferno')

sns.move_legend(
    ax8, "lower center",
    bbox_to_anchor=(0.5, -.2),
    ncol=3,
    frameon=False,
)

plt.title('Total Number of Customers by Month')
plt.xlabel('Month of Operation')
plt.ylabel('Total Number of Customers')
plt.ylim((0, 160))

plt.axvspan(*mdates.datestr2num(['2019-02-01', '2019-04-01']), color = 'grey')
plt.grid()

```



The trial period is highlighted by the grey strip

Total number of customers in the trial period for the trial store is significantly higher than the control store or two out of three months, which indicates a positive trial effect.

## Conclusion

- We've found control stores 233, 155, 237 for trial stores 77, 86 and 88 respectively.
- The results for trial stores 77 and 88 during the trial period show a significant difference in at least two of the three trial months but this is not the case for trial store 86. We can check with the client if the implementation of the trial was different in trial store 86 but overall, the trial shows a significant increase in sales.
- Now that we have finished our analysis, we can prepare our presentation to the Category Manager.